Metropolis

Contents

Notation

An overdensity vector contains signal plus photometric zero-point noise plus shot noise. It can be represented in three different bases as listed below,

$$\boldsymbol{\Gamma} = \boldsymbol{\kappa} + \boldsymbol{\eta} + \boldsymbol{\zeta}, \quad \textit{cell space}$$

$$\boldsymbol{m} = \boldsymbol{s} + \boldsymbol{n} + \boldsymbol{z}, \quad \textit{signal space}$$

$$\boldsymbol{\mathscr{m}} = \boldsymbol{\mathscr{s}} + \boldsymbol{\mathscr{n}} + \boldsymbol{\mathscr{z}}, \quad W\ \textit{space}$$

I will occasionally work with the noise terms as a unit using $\boldsymbol{\nu} \equiv \boldsymbol{\eta} + \boldsymbol{\zeta}$ and $\boldsymbol{\mathscr{v}} \equiv \boldsymbol{\mathscr{n}} + \boldsymbol{\mathscr{z}}$.

We can rotate between cell-space and either W-space or signal-space using an eigenvector matrix **W** or **Z** respectively,

$$\boldsymbol{\mathscr{m}} = \mathbf{W}^T\boldsymbol{\Gamma}, \qquad \boldsymbol{\Gamma} = \mathbf{W}\boldsymbol{\mathscr{m}},$$
$$\boldsymbol{m} = \mathbf{Z}^T\boldsymbol{\Gamma}, \qquad \boldsymbol{\Gamma} = \mathbf{Z}\boldsymbol{m},$$

W results from the diagonalization of the following sum of inverses of the signal and noise covariance matrices,

$$\boldsymbol{\Sigma}_\kappa^{-1} + \boldsymbol{\Sigma}_\nu^{-1} = \mathbf{W}\boldsymbol{\Lambda}^{(\kappa\nu)}\mathbf{W}^T,$$

where

$$\mathbf{\Sigma}_\nu^{-1} = \left(\mathbf{\Sigma}_\eta + \mathbf{\Sigma}_\zeta\right)^{-\mathbf{1}}.$$

I distinguish between simulated overdensity realizations using superscripts. For example, the cell-space vector $\boldsymbol{\Gamma}^{(\alpha)} = \boldsymbol{\kappa}^{(\alpha)} + \boldsymbol{\eta}^{(\alpha)} + \boldsymbol{\zeta}^{(\alpha)}$ corresponds to $\boldsymbol{m}^{(\alpha)} = \boldsymbol{s}^{(\alpha)} + \boldsymbol{n}^{(\alpha)} + \boldsymbol{z}^{(\alpha)}$ in W-space components. Signal, zero-point noise and shot noise can be fully specified for simulated cases.

Photometric zero-point noise realizations are generated in *segment-space*. There are 2052 SEGMENTs in DR6, each one of which has a i.i.d. zero-point offset $\Delta m \sim \mathcal{N}(0, \sigma_m^2)$. The covariance matrix of the zero-point offsets in segment-space $\mathbf{\Sigma}_{\Delta m}$ is therefore diagonal with every non-zero element equal to ${\sigma_m}^2$. There is a mapping matrix $\mathbf{A}$ that rotates the overdensities due to the zero-points back into cell-space, $\boldsymbol{\eta} = \mathbf{A} \cdot \Delta \boldsymbol{m}$ such that $\mathbf{\Sigma}_\eta = {\sigma_m}^2 \mathbf{A}\mathbf{A}^T$.

Shot noise is diagonal in cell-space where $\zeta_i \sim \mathcal{N}(0, 1/\langle n_i \rangle)$ and $\langle n_i \rangle$ is the number of galaxies expected in the i^{th} cell. This value is determined from the galaxy selection function.

Instances of $\boldsymbol{\kappa}$, $\boldsymbol{\eta}$, and $\boldsymbol{\zeta}$ are each the result of a Gaussian random process and therefore can be parameterized in terms of the Gaussian variates that comprise them. I will represent this parameterized format as $\boldsymbol{\Gamma}(\boldsymbol{p}) = \boldsymbol{\kappa}(\boldsymbol{\theta}) + \boldsymbol{\eta}(\boldsymbol{\varphi}) + \boldsymbol{\zeta}(\boldsymbol{\gamma})$ where the full parameterization vector $\boldsymbol{p} = \{\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\gamma}\}$. The parameters can also be grouped into subsets. One convenient representation, as we will see, is $\boldsymbol{\Gamma}(\boldsymbol{p}) = \boldsymbol{\kappa}(\boldsymbol{\theta}) + \boldsymbol{v}(\boldsymbol{\psi})$ where $\boldsymbol{\psi} = \{\boldsymbol{\varphi}, \boldsymbol{\gamma}\}$.

However, the universe contains only one *actual* overdensity vector for which I reserve the symbols $\boldsymbol{\delta}, \boldsymbol{d}$ and $\boldsymbol{d}$ for cell, signal and W-spaces respectively,

$$\boldsymbol{\delta} = \boldsymbol{\delta}_\kappa + \boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta,$$
$$\boldsymbol{d} = \boldsymbol{d}_{\boldsymbol{s}} + \boldsymbol{d}_{\boldsymbol{n}} + \boldsymbol{d}_{\boldsymbol{z}},$$
$$\boldsymbol{d} = \boldsymbol{d}_{\boldsymbol{s}} + \boldsymbol{d}_{\boldsymbol{n}} + \boldsymbol{d}_{\boldsymbol{z}}.$$

In the real world, only the left-hand sides of the equations are known. When we are running diagnostics, all terms can be specified.

Signal Prediction Through Bayes's Theorem

The expected value of the i^{th} signal coefficient can be expressed as a function of the posterior probability of a partial overdensity model $\boldsymbol{\theta}$ given the data,

$$\langle \kappa_i | \boldsymbol{\delta} \rangle = \int \kappa_i(\boldsymbol{\theta})\, P(\boldsymbol{\theta} | \boldsymbol{\delta})\, d\boldsymbol{\theta}.$$

If I let $\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}$ represent random variates drawn from the distribution $P(\boldsymbol{\theta}|\boldsymbol{\delta})$, the estimated signal given the data is, through a Monte Carlo process,

$$\hat{\kappa}_i = \frac{1}{K}\sum_{\alpha=1}^{K} \kappa_i\big(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\big),$$

in cell-space and

$$\hat{\mathscr{s}}_i = \frac{1}{K}\sum_{\alpha=1}^{K} \mathscr{s}_i\big(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\big),$$

in W-space. It should be the case that

$$\langle \kappa_i|\boldsymbol{\delta}\rangle = \lim_{K\to\infty} \hat{\kappa}_i.$$

There is no immediately obvious way to generate variables from $P(\boldsymbol{\theta}|\boldsymbol{\delta})$ directly, so I invoke Bayes's theorem,

$$P(\boldsymbol{\theta}|\boldsymbol{\delta}) = \frac{P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{\delta})},$$

where

$$P(\boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T\boldsymbol{\Sigma}_\kappa^{-1}\boldsymbol{\kappa}(\boldsymbol{\theta})\right] = \exp\left[-\frac{1}{2}\boldsymbol{\mathscr{s}}(\boldsymbol{\theta})^T\boldsymbol{\Sigma}_{\mathscr{s}}^{-1}\boldsymbol{\mathscr{s}}(\boldsymbol{\theta})\right],$$

and

$$P(\boldsymbol{\delta}|\boldsymbol{\theta}) \propto \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta}-\boldsymbol{\kappa}(\boldsymbol{\theta})\big)^T\boldsymbol{\Sigma}_\nu^{-1}\big(\boldsymbol{\delta}-\boldsymbol{\kappa}(\boldsymbol{\theta})\big)\right] = \exp\left[-\frac{1}{2}\big(\boldsymbol{d}-\boldsymbol{\mathscr{s}}(\boldsymbol{\theta})\big)^T\boldsymbol{\Sigma}_{\mathscr{v}}^{-1}\big(\boldsymbol{d}-\boldsymbol{\mathscr{s}}(\boldsymbol{\theta})\big)\right].$$

Multiplying and expanding,

$$\begin{aligned}P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta}) &\propto \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta}-\boldsymbol{\kappa}(\boldsymbol{\theta})\big)^T\boldsymbol{\Sigma}_\nu^{-1}\big(\boldsymbol{\delta}-\boldsymbol{\kappa}(\boldsymbol{\theta})\big)\right]\exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T\boldsymbol{\Sigma}_\kappa^{-1}\boldsymbol{\kappa}(\boldsymbol{\theta})\right]\\ &\propto \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\delta} - 2\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\kappa}(\boldsymbol{\theta}) + \boldsymbol{\kappa}(\boldsymbol{\theta})^T(\boldsymbol{\Sigma}_\kappa^{-1}+\boldsymbol{\Sigma}_\nu^{-1})\boldsymbol{\kappa}(\boldsymbol{\theta})\big)\right].\end{aligned}$$

The term $\exp\left[-\frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\delta}\right]$ is constant and can be disregarded in this proportion. $\boldsymbol{\mathscr{s}} = \mathbf{W}^T\boldsymbol{\kappa}$ and $\mathbf{W}$ is orthonormal, therefore

$$\begin{aligned}P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta}) &\propto \exp[\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\kappa}(\boldsymbol{\theta})]\exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T(\boldsymbol{\Sigma}_\kappa^{-1}+\boldsymbol{\Sigma}_\nu^{-1})\boldsymbol{\kappa}(\boldsymbol{\theta})\right]\\ &\propto \exp[\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}\mathbf{W}^T\boldsymbol{\kappa}(\boldsymbol{\theta})]\exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T\mathbf{W}\mathbf{W}^T(\boldsymbol{\Sigma}_\kappa^{-1}+\boldsymbol{\Sigma}_\nu^{-1})\mathbf{W}\mathbf{W}^T\boldsymbol{\kappa}(\boldsymbol{\theta})\right]\end{aligned}$$

$$\propto \exp[\mathbf{x}^T \boldsymbol{\mathcal{s}}(\boldsymbol{\theta})] \exp\left[-\frac{1}{2}\boldsymbol{\mathcal{s}}(\boldsymbol{\theta})^T \boldsymbol{\Lambda}^{(\kappa\nu)} \boldsymbol{\mathcal{s}}(\boldsymbol{\theta})\right],$$

where $\mathbf{x}^T = \boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \mathbf{W}$.

$\boldsymbol{\mathcal{s}}(\boldsymbol{\theta})$ is an N-dimensional vector with N degrees of freedom. We have the freedom to choose any N parameters θ that most conveniently map $\boldsymbol{\theta} \rightarrow \boldsymbol{\mathcal{s}}$. A natural choice is $\theta_i = \mathcal{s}_i$ for all i.

We note that number of degrees of $\boldsymbol{v}(\boldsymbol{\varphi}, \boldsymbol{\gamma})$ is $N + 2052$ while the dimensionality of W-space is only N, a fact that might suggest information is being discarded. However, both the shot noise and zero-point noise processes can be entirely represented within N dimensional cell-space. Consequently, their *sum* can also be expressed in N dimensions. Since we do not deal with the underlying degeneracies between these two types of noise, combining them in this fashion will not impede our estimation of the N signal parameters $\boldsymbol{\theta}$.

Analytic W-Space Solution

Linear Estimator

The expected value of the ith signal coefficient can be expressed as a function of the posterior probability of a partial overdensity model $\boldsymbol{\theta}$ given the data,

$$\langle \kappa_i | \boldsymbol{\delta} \rangle = \int \kappa_i(\boldsymbol{\theta})\, P(\boldsymbol{\theta}|\boldsymbol{\delta})\, d\boldsymbol{\theta}.$$

I invoke Bayes's theorem,

$$P(\boldsymbol{\theta}|\boldsymbol{\delta}) = \frac{P(\boldsymbol{\delta}|\boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\boldsymbol{\delta})},$$

where

$$P(\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_\kappa|}} \exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T \boldsymbol{\Sigma}_\kappa^{-1} \boldsymbol{\kappa}(\boldsymbol{\theta})\right],$$

and

$$P(\boldsymbol{\delta}|\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}_\nu|}} \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta} - \boldsymbol{\kappa}(\boldsymbol{\theta})\big)^T \boldsymbol{\Sigma}_\nu^{-1} \big(\boldsymbol{\delta} - \boldsymbol{\kappa}(\boldsymbol{\theta})\big)\right].$$

Multiplying and expanding,

$$P(\boldsymbol{\delta}|\boldsymbol{\theta}) P(\boldsymbol{\theta}) = \frac{1}{(2\pi)^N \sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}} \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta} - \boldsymbol{\kappa}(\boldsymbol{\theta})\big)^T \boldsymbol{\Sigma}_\nu^{-1} \big(\boldsymbol{\delta} - \boldsymbol{\kappa}(\boldsymbol{\theta})\big)\right] \exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T \boldsymbol{\Sigma}_\kappa^{-1} \boldsymbol{\kappa}(\boldsymbol{\theta})\right]$$

$$= \frac{1}{(2\pi)^N \sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}} \exp\left[-\frac{1}{2}\left(\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\delta} - 2\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\kappa}(\boldsymbol{\theta}) + \boldsymbol{\kappa}(\boldsymbol{\theta})^T (\boldsymbol{\Sigma}_\kappa^{-1} + \boldsymbol{\Sigma}_\nu^{-1}) \boldsymbol{\kappa}(\boldsymbol{\theta})\right)\right].$$

$\boldsymbol{s} = \mathbf{W}^T \boldsymbol{\kappa}$ and $\mathbf{W}$ is orthonormal, therefore

$$P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta}) = \frac{1}{(2\pi)^N \sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}} \exp\left[-\frac{1}{2}\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\delta}\right] \exp[\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\kappa}(\boldsymbol{\theta})] \exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T (\boldsymbol{\Sigma}_\kappa^{-1} + \boldsymbol{\Sigma}_\nu^{-1}) \boldsymbol{\kappa}(\boldsymbol{\theta})\right]$$

$$= \frac{\exp\left[-\frac{1}{2}\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\delta}\right]}{(2\pi)^N \sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}} \exp[\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \mathbf{W}\mathbf{W}^T \boldsymbol{\kappa}(\boldsymbol{\theta})] \exp\left[-\frac{1}{2}\boldsymbol{\kappa}(\boldsymbol{\theta})^T \mathbf{W}\mathbf{W}^T (\boldsymbol{\Sigma}_\kappa^{-1} + \boldsymbol{\Sigma}_\nu^{-1}) \mathbf{W}\mathbf{W}^T \boldsymbol{\kappa}(\boldsymbol{\theta})\right]$$

$$= \frac{\exp\left[-\frac{1}{2}\boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \boldsymbol{\delta}\right]}{(2\pi)^N \sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}} \exp\left[\mathbf{x}^T \boldsymbol{s}(\boldsymbol{\theta}) - \frac{1}{2}\boldsymbol{s}(\boldsymbol{\theta})^T \boldsymbol{\Lambda}^{(\kappa\nu)} \boldsymbol{s}(\boldsymbol{\theta})\right],$$

where $\mathbf{x}^T = \boldsymbol{\delta}^T \boldsymbol{\Sigma}_\nu^{-1} \mathbf{W}$.

To normalize this function I note that $\int P(\boldsymbol{\theta}|\boldsymbol{\delta})\, d\boldsymbol{\theta} = 1$ and therefore

$$\int \frac{P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{\delta})}\, d\boldsymbol{\theta} = 1,$$

and

$$P(\boldsymbol{\delta}) = \int P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta})\, d\boldsymbol{\theta}.$$

Given that the signal we seek to estimate is N-dimensional, we have the freedom to choose any N parameters that most conveniently map $\boldsymbol{\theta} \rightarrow \boldsymbol{\kappa}$ or $\boldsymbol{\theta} \rightarrow \boldsymbol{s}$. The form of the probability distribution function suggests a natural choice of $\theta_i = s_i$ for all i. From this perspective, it is more convenient to solve the problem

$$\langle s_i|\boldsymbol{\delta}\rangle = \int s_i\, P(\boldsymbol{s}|\boldsymbol{\delta})\, d\boldsymbol{s}.$$

Because of the diagonality of $\boldsymbol{\Lambda}^{(\kappa\nu)}$, the expression for $P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta})$ is separable in the exponent,

$$\langle s_i|\boldsymbol{\delta}\rangle = \int s_i\, P(\boldsymbol{s}|\boldsymbol{\delta})\, d\boldsymbol{s}$$

$$= \frac{\int s_i\, P(\boldsymbol{\delta}|\boldsymbol{s})P(\boldsymbol{s})\, d\boldsymbol{s}}{\int P(\boldsymbol{\delta}|\boldsymbol{s}')P(\boldsymbol{s}')\, d\boldsymbol{s}'}$$

$$= \frac{\int \mathscr{s}_i \frac{\exp\left[-\frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\delta}\right]}{(2\pi)^N\sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}}\exp\left[\mathbf{x}^T\boldsymbol{\mathscr{s}} - \frac{1}{2}\boldsymbol{\mathscr{s}}^T\boldsymbol{\Lambda}^{(\kappa\nu)}\boldsymbol{\mathscr{s}}\right]\, d\boldsymbol{\mathscr{s}}}{\int \frac{\exp\left[-\frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\delta}\right]}{(2\pi)^N\sqrt{|\boldsymbol{\Sigma}_\kappa||\boldsymbol{\Sigma}_\nu|}}\exp\left[\mathbf{x}^T\boldsymbol{\mathscr{s}}' - \frac{1}{2}\boldsymbol{\mathscr{s}}'^T\boldsymbol{\Lambda}^{(\kappa\nu)}\boldsymbol{\mathscr{s}}'\right] d\boldsymbol{\mathscr{s}}'}$$

$$= \frac{\int \mathscr{s}_i\ \exp\left[\mathbf{x}^T\boldsymbol{\mathscr{s}} - \frac{1}{2}\boldsymbol{\mathscr{s}}^T\boldsymbol{\Lambda}^{(\kappa\nu)}\boldsymbol{\mathscr{s}}\right]\, d\boldsymbol{\mathscr{s}}}{\int \exp\left[\mathbf{x}^T\boldsymbol{\mathscr{s}}' - \frac{1}{2}\boldsymbol{\mathscr{s}}'^T\boldsymbol{\Lambda}^{(\kappa\nu)}\boldsymbol{\mathscr{s}}'\right] d\boldsymbol{\mathscr{s}}'}$$

$$= \frac{\int \mathscr{s}_i\ \exp\left[\sum_{j=1}^N\left(x_j\mathscr{s}_j - \frac{1}{2}\mathscr{s}_j^2\lambda_j^{(\kappa\nu)}\right)\right]\, d\mathscr{s}_1 d\mathscr{s}_2\cdots d\mathscr{s}_N}{\int \exp\left[\sum_{j=1}^N\left(x_j\mathscr{s}_j' - \frac{1}{2}\mathscr{s}_j'^2\lambda_j^{(\kappa\nu)}\right)\right]\, d\mathscr{s}_1' d\mathscr{s}_2'\cdots d\mathscr{s}_N'}$$

$$= \frac{\int \mathscr{s}_i\ \prod_{j=1}^N \exp\left[x_j\mathscr{s}_j - \frac{1}{2}\mathscr{s}_j^2\lambda_j^{(\kappa\nu)}\right] d\mathscr{s}_j}{\int\ \prod_{j=1}^N \exp\left[x_j\mathscr{s}_j' - \frac{1}{2}\mathscr{s}_j'^2\lambda_j^{(\kappa\nu)}\right] d\mathscr{s}_j'}.$$

All but one of the integrals in the numerator and denominator will cancel,

$$\langle \mathscr{s}_i|\boldsymbol{\delta}\rangle = \frac{\int \exp\left[x_1\mathscr{s}_1 - \frac{1}{2}\mathscr{s}_1^2\lambda_1^{(\kappa\nu)}\right] d\mathscr{s}_1 \cdots \int \mathscr{s}_i \exp\left[x_i\mathscr{s}_i - \frac{1}{2}\mathscr{s}_i^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i \cdots \int \exp\left[x_N\mathscr{s}_N - \frac{1}{2}\mathscr{s}_N^2\lambda_N^{(\kappa\nu)}\right] d\mathscr{s}_N}{\int \exp\left[x_1\mathscr{s}_1' - \frac{1}{2}\mathscr{s}_1'^2\lambda_1^{(\kappa\nu)}\right] d\mathscr{s}_1' \cdots \int \exp\left[x_i\mathscr{s}_i' - \frac{1}{2}\mathscr{s}_i'^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i' \cdots \int \exp\left[x_N\mathscr{s}_N' - \frac{1}{2}\mathscr{s}_N'^2\lambda_N^{(\kappa\nu)}\right] d\mathscr{s}_N'},$$

$$\langle \mathscr{s}_i|\boldsymbol{\delta}\rangle = \frac{\int \mathscr{s}_i \exp\left[x_i\mathscr{s}_i - \frac{1}{2}\mathscr{s}_i^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i}{\int \exp\left[x_i\mathscr{s}_i' - \frac{1}{2}\mathscr{s}_i'^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i'}.$$

We integrate over all possible values of $\mathscr{s}_i$ from $-\infty$ to ∞.

$$\langle \mathscr{s}_i|\boldsymbol{\delta}\rangle = \frac{\int_{-\infty}^{\infty} \mathscr{s}_i \exp\left[x_i\mathscr{s}_i - \frac{1}{2}\mathscr{s}_i^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i}{\int_{-\infty}^{\infty} \exp\left[x_i\mathscr{s}_i' - \frac{1}{2}\mathscr{s}_i'^2\lambda_i^{(\kappa\nu)}\right] d\mathscr{s}_i'}$$

$$= \frac{x_i}{\lambda_i^{(\kappa\nu)}}.$$

Once all values of $\hat{\boldsymbol{\mathscr{s}}}$ are evaluated, the solution in cell-space can be found through one final rotation $\langle \boldsymbol{\kappa}|\boldsymbol{\delta}\rangle = \mathbf{W} * \langle \boldsymbol{\mathscr{s}}|\boldsymbol{\delta}\rangle$. It should also be clear that

$$\langle s_i s_j | \boldsymbol{\delta} \rangle = \frac{x_i}{\lambda_i^{(\kappa\nu)}} \frac{x_j}{\lambda_j^{(\kappa\nu)}}.$$

Likewise, the expected value of the signal squared can be found through integration,

$$\langle s_i^2 | \boldsymbol{\delta} \rangle = \frac{\int_{-\infty}^{\infty} s_i^2 \exp\left[x_i s_i - \frac{1}{2} s_i^2 \lambda_i^{(\kappa\nu)}\right] ds_i}{\int_{-\infty}^{\infty} \exp\left[x_i s_i' - \frac{1}{2} s_i'^2 \lambda_i^{(\kappa\nu)}\right] ds_i'}$$

$$= \frac{x_i^2 + \lambda_i^{(\kappa\nu)}}{\lambda_i^{(\kappa\nu)2}}.$$

This implies that the variance of the solution in W-space is

$$\mathrm{Var}(s_\mathrm{i}|\delta) = \langle s_\mathrm{i}^2|\delta\rangle - \langle s_\mathrm{i}|\delta\rangle^2 = \frac{1}{\lambda_\mathrm{i}^{(\kappa\nu)}}.$$

From my observations of the R7, R11 and R16 cases $\lambda_\mathrm{i}^{(\kappa\nu)} > 1 \;\forall\; i$. It is also interesting to note that the variance of the solution is independent of $\boldsymbol{\delta}$. I assume that if the variance is independent of the data in one space, it must be in all the other spaces as well.

$\boldsymbol{\delta}$	`clean/shot/delta_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\boldsymbol{\lambda}^{(\kappa\nu)}$	`clean/Eigen_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Values_fwriteR`
$\mathbf{W}$	`clean/Eigen_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Vectors_fwriteC`
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	`t…/clean/Sigma_nu_inv_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteC`
clean/MHK.m ~ 15 sec for 200 vectors, R8	
$\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle$	`clean/shot/kgd_R[`r`]_sp[02]_[`N_{realz}`]_fwriteC`

This solution is general to any Gaussian signal and noise and is applicable for any systematic pattern on the sky.

As a sanity check, we examined the covariances between $\kappa_i|\boldsymbol{\delta}$. We expect that the κ_i for cells intersecting the same SEGMENT should be more positively correlated with each other than with the κ_i values corresponding to cells that do not. We selected the longest SEGMENT in our simulation (the one corresponding to A's 2050th column) and found that it intersects 2019 cells. One of those cells (the one corresponding to A's 534th row) was selected at random.

Using 10,000 $\boldsymbol{\delta}$ realizations, we calculated $\mathrm{Cov}(\kappa_i|\boldsymbol{\delta}, \kappa_i|\boldsymbol{\delta})$ for each of the 2019 cells that share a SEGMENT with #534. We found that the average covariance was 0.0020 with a standard deviation of

0.027. The average covariance between cell #534 with the cells that don't share the same SEGMENT was $7.5e-5$ with a standard deviation of 0.010. I repeated calculation for cell #881 and found a covariance of 0.0014 ± 0.03 for cells intersecting the same SEGMENT and $-7.7e-5 \pm 0.01$ for cells that do not.

While this is admittedly a small measurement sample, it provides evidence that the estimated signal in cell-space has a covariance structure linked to the cell/SEGMENT intersections. (I could have extended this, but it would have been time consuming and I don't know whether it's worthy of publication in any event.)

Quadratic Estimator

We can also solve for the expected signal squared in W-space and in signal-space. If we define $\mathbf{M} \equiv \mathbf{Z}^T\mathbf{W}$, then $\boldsymbol{s} = \mathbf{M}\boldsymbol{\mathscr{s}}$. For a particular element

$$s_i = \sum \mathrm{M}[i,:].*\,\boldsymbol{\mathscr{s}} = M_{i,1}\mathscr{s}_1 + M_{i,2}\mathscr{s}_2 + \cdots + M_{i,N}\mathscr{s}_N.$$

W	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
Z	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
SmallCode.m (J) ~ (284/56/18/1) minutes for R6/R7/R8/R16	
M	g…/clean/M[r]_[sp02]_[z_{min}]_[z_{max}]_[bp62]_fwriteC

The signal coefficient squared in signal-space is found through a double sum over the signal coefficients in W-space,

$$s_i^2 = \sum_l \sum_m M_{i,l} M_{i,m}\, \mathscr{s}_l \mathscr{s}_m$$

$$= \sum_{\substack{l,m \\ l\neq m}} M_{i,l} M_{i,m}\, \mathscr{s}_l \mathscr{s}_m + \sum_m M_{i,m}^2\, \mathscr{s}_m^2.$$

I'm not 100% confident that this next step is true, but let's argue that

$$\langle s_i^2|\boldsymbol{\delta}\rangle = \sum_{\substack{l,m \\ l\neq m}} M_{i,l} M_{i,m}\, \langle \mathscr{s}_l \mathscr{s}_m|\boldsymbol{\delta}\rangle + \sum_m M_{i,m}^2\, \langle \mathscr{s}_m^2|\boldsymbol{\delta}\rangle.$$

I'll try to rearrange to make the computation more efficient,

$$\langle s_i^2|\boldsymbol{\delta}\rangle = \sum_{\substack{l,m \\ l\neq m}} M_{i,l}\langle \mathscr{s}_l|\boldsymbol{\delta}\rangle\, M_{i,m}\langle \mathscr{s}_m|\boldsymbol{\delta}\rangle + \sum_m M_{i,m}^2\, \langle \mathscr{s}_m^2|\boldsymbol{\delta}\rangle$$

$$= \sum_{l,m} M_{i,l}\langle \mathscr{s}_l|\boldsymbol{\delta}\rangle\, M_{i,m}\langle \mathscr{s}_m|\boldsymbol{\delta}\rangle - \sum_m M_{i,m}^2 \langle \mathscr{s}_m|\boldsymbol{\delta}\rangle\langle \mathscr{s}_m|\boldsymbol{\delta}\rangle + \sum_m M_{i,m}^2\, \langle \mathscr{s}_m^2|\boldsymbol{\delta}\rangle$$

$$= \sum_{l,m} M_{i,l}\langle \mathscr{s}_l|\boldsymbol{\delta}\rangle\, M_{i,m}\langle \mathscr{s}_m|\boldsymbol{\delta}\rangle + \sum_m M_{i,m}^2\, \left(\langle \mathscr{s}_m^2|\boldsymbol{\delta}\rangle - \langle \mathscr{s}_m|\boldsymbol{\delta}\rangle\langle \mathscr{s}_m|\boldsymbol{\delta}\rangle\right)$$

$$= \sum_{l,m} M_{i,l}\langle \mathcal{s}_l|\boldsymbol{\delta}\rangle\, M_{i,m}\langle \mathcal{s}_m|\boldsymbol{\delta}\rangle + \sum_m M_{i,m}^2 \left(\frac{x_m^2 + \lambda_m^{(\kappa\nu)}}{{\lambda_m^{(\kappa\nu)}}^2} - \frac{x_m}{\lambda_m^{(\kappa\nu)}} \frac{x_m}{\lambda_m^{(\kappa\nu)}} \right)$$

$$= \sum_{l,m} M_{i,l}\langle \mathcal{s}_l|\boldsymbol{\delta}\rangle\, M_{i,m}\langle \mathcal{s}_m|\boldsymbol{\delta}\rangle + \sum_m \frac{M_{i,m}^2}{\lambda_m^{(\kappa\nu)}}.$$

I can define a quantity $g_i \equiv \sum_l M_{i,l}\langle \mathcal{s}_l|\boldsymbol{\delta}\rangle$ or $\boldsymbol{g} = \mathbf{M}\langle \boldsymbol{\mathcal{s}}|\boldsymbol{\delta}\rangle$. Then,

$$\langle s_i^2|\boldsymbol{\delta}\rangle = g_i^2 + \sum_m \frac{M_{i,m}^2}{\lambda_m^{(\kappa\nu)}}.$$

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{M}$	g…/clean/M[r]_[sp02]_[z_{min}]_[z_{max}]_[bp62]_fwriteC
clean/MHKs2.m ~ 10,000 vectors in (17/8/1) minutes for R7/R8/R16	
$\langle \boldsymbol{s}^2|\boldsymbol{\delta}\rangle$	clean/shot/s2gd_R[r]_sp[02]_[N_{realz}]_fwriteC

Sampling $P(\theta|\delta)$ with Metropolis-Hastings

The Metropolis-Hastings algorithm is a Markov chain Monte Carlo method for generating realizations from a distribution that is difficult to sample directly. To use Metropolis-Hastings it is sufficient to have a function that is *proportional to* that distribution. This is certainly the case with this problem. I wish to sample $P(\boldsymbol{\theta}|\boldsymbol{\delta})$ and possess an analytic form for $P(\boldsymbol{\delta}|\boldsymbol{\theta})P(\boldsymbol{\theta}) \propto P(\boldsymbol{\theta}|\boldsymbol{\delta})$.

The Metropolis-Hastings Method

To simplify notation, let the unscaled posterior density be denoted

$$g(\boldsymbol{\theta}|\boldsymbol{\delta}) = \exp[\mathbf{x}^T \boldsymbol{\mathcal{s}}(\boldsymbol{\theta})] \exp\left[-\frac{1}{2}\boldsymbol{\mathcal{s}}(\boldsymbol{\theta})^T \boldsymbol{\Lambda}^{(\kappa\nu)} \boldsymbol{\mathcal{s}}(\boldsymbol{\theta})\right].$$

Let us also introduce an *independent candidate density* $q(\boldsymbol{\theta})$ that will ideally match the correlation structure of $g(\boldsymbol{\theta}|\boldsymbol{\delta})$ but with a broader tail so that the former "blankets" the latter. I will derive the exact form of this distribution in the next section.

The steps for the Metropolis-Hastings method are as follows:

1. Choose an initial set of signal parameters $\boldsymbol{\theta}^{(0)}$ that preferably lies near the peak of $g(\boldsymbol{\theta}|\boldsymbol{\delta})$.

2. Repeat the following steps n times for $i = 1, \ldots, n$:
   a) Draw a new random vector $\boldsymbol{\theta}'$ from $q(\boldsymbol{\theta})$.
   b) Calculate the *acceptance probability* $\alpha\left(\boldsymbol{\theta}^{(i-1)}|\boldsymbol{\theta}'\right)$ where

   $$\alpha(\boldsymbol{\theta}|\boldsymbol{\theta}') = \min\left[1, \frac{g(\boldsymbol{\theta}'|\boldsymbol{\delta})q(\boldsymbol{\theta})}{g(\boldsymbol{\theta}|\boldsymbol{\delta})q(\boldsymbol{\theta}')}\right].$$

   c) Draw a random variate u from the uniform distribution $U(0,1)$.
   d) If $u < \alpha\left(\boldsymbol{\theta}^{(i-1)}|\boldsymbol{\theta}'\right)$, then set $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}'$ and accept $\boldsymbol{\theta}'$ as one of the random variates of $P(\boldsymbol{\theta}|\boldsymbol{\delta})$. Otherwise, set $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)}$.

Independent Candidate Density

We select an independent candidate density $q(\boldsymbol{\theta})$ that has approximately the same shape as the target distribution $g(\boldsymbol{\theta}|\boldsymbol{\delta})$. A reasonable first guess is a multivariate Gaussian distribution that peaks in the same place as $g(\boldsymbol{\theta}|\boldsymbol{\delta})$ - let's call it $\widehat{\boldsymbol{\theta}}$ - and has the same curvature as $g(\boldsymbol{\theta}|\boldsymbol{\delta})$ at $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$. However, in practice the Gaussian is usually not broad enough to provide adequate coverage at the tails of $g(\boldsymbol{\theta}|\boldsymbol{\delta})$.

A better candidate density is the multivariate t-distribution $t_f\left(\widehat{\boldsymbol{\theta}}, \mathbf{Y}\right)$. This density function should have the same peak and curvature at the peak as the Gaussian, but at low degrees of freedom f it is more adequate for representative sampling. (As f increases, t_f thins and asymptotically approaches the multivariate Gaussian distribution – we do not want to go too far in this direction.)

To find the peak and curvature we begin with the logarithm of the target density,

$$l(\boldsymbol{\theta}|\boldsymbol{\delta}) = \ln\left(g(\boldsymbol{\theta}|\boldsymbol{\delta})\right) = \mathbf{x}^T \boldsymbol{s}(\boldsymbol{\theta}) - \frac{1}{2}\boldsymbol{s}(\boldsymbol{\theta})^T \boldsymbol{\Lambda}^{(\kappa\nu)} \boldsymbol{s}(\boldsymbol{\theta}).$$

I note that a function and its logarithm have maxima located at the same position. The first derivative of $l(\boldsymbol{\theta}|\boldsymbol{\delta})$ where $\theta_i = s_i$ is

$$\begin{pmatrix} \dfrac{\partial l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_N} \end{pmatrix} = \begin{pmatrix} x_1 - \lambda_1^{(\kappa\nu)} s_1 \\ \vdots \\ x_N - \lambda_N^{(\kappa\nu)} s_N \end{pmatrix},$$

Setting this equal to zero reveals the location of the target density's maximum,

$$\widehat{\boldsymbol{\theta}} = \begin{pmatrix} x_1/\lambda_1^{(\kappa\nu)} \\ \vdots \\ x_N/\lambda_N^{(\kappa\nu)} \end{pmatrix}.$$

In practice, $\langle x_i \rangle \approx 0$, $\langle |x_i| \rangle \approx 8$ and $\mathrm{Var}(x) \approx 270$.

The inverse of the target density's second derivative at $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$ yields the curvature of a multivariate normal that peaks at the same location

$$\mathbf{Y} = -\begin{bmatrix} \dfrac{\partial^2 l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_1^2} & \cdots & \dfrac{\partial^2 l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_1 \partial \theta_N} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_N \partial \theta_1} & \cdots & \dfrac{\partial^2 l(\boldsymbol{\theta}|\boldsymbol{\delta})}{\partial \theta_N^2} \end{bmatrix}_{\boldsymbol{\theta}=\widehat{\boldsymbol{\theta}}}^{\mathbf{-1}} = -\begin{bmatrix} -\lambda_1^{(\kappa\nu)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -\lambda_N^{(\kappa\nu)} \end{bmatrix}^{-1}.$$

This simplifies to $\mathbf{Y} = \boldsymbol{\Lambda}^{(\kappa\nu)^{\mathbf{-1}}}$.

Dropping the constant terms (since they cancel anyway), my independent candidate distribution adopts the form of $t_f\left(\widehat{\boldsymbol{\theta}}, \mathbf{Y}\right)$,

$$q(\boldsymbol{\theta}) = \left(1 + \frac{1}{f}\left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right)^T \boldsymbol{\Lambda}^{(\kappa\nu)}\left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right)\right)^{-(N+f)/2}.$$

To sample $\boldsymbol{\theta}'$ from this distribution, I begin by solving for the lower triangular matrix $\mathbf{L}$ that satisfies the Cholesky decomposition $\mathbf{Y} = \mathbf{L}\mathbf{L}^T$. The diagonality of $\mathbf{Y}$ yields a simple solution,

$$\mathbf{L} = \begin{bmatrix} 1/\sqrt{\lambda_1^{(\kappa\nu)}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\sqrt{\lambda_N^{(\kappa\nu)}} \end{bmatrix}.$$

When a set of random variables is drawn from the t-distribution with $f = 1$,

$$\boldsymbol{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_N \end{pmatrix},$$

the vector

$$\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}} + \mathbf{L}\boldsymbol{t}$$

will constitute a random draw from $q(\boldsymbol{\theta})$.

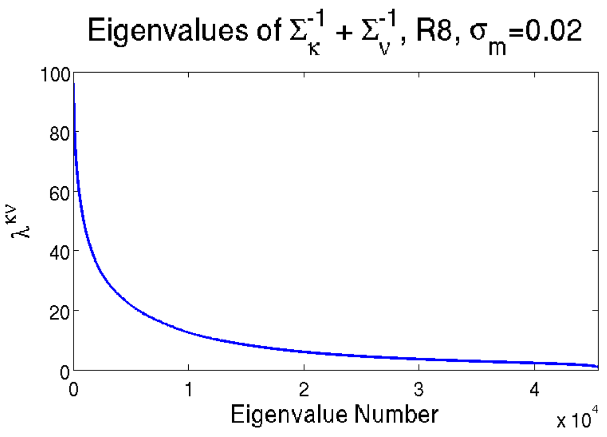
W-Space

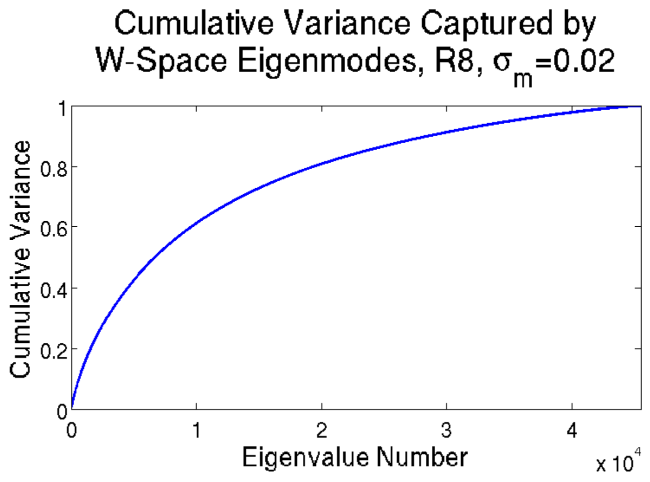
It’s predictably time-consuming to generate two inverses and the diagonalization of their sum. Be sure to forcibly symmetrize the matrices before inversion or diagonalization. Otherwise small numerical discrepancies will cause the processes to take forever.

$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \lvert\mathbf{d}\rvert, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n \rangle, \delta$	base/Overdensities_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR
$\boldsymbol{\Sigma_\kappa}$	signal/Sigma_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{A}$	cells_grids/A_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR
clean/Wspace.m ~ 5.3hr for $\boldsymbol{\Sigma_\nu^{-1}}$, 19hr for $\boldsymbol{\Sigma_\kappa^{-1}}$, 11hr (29.5hr – R7) for diagonalization, 15min for $\boldsymbol{\Sigma_\nu^{-1}\mathbf{W}}$ (R8)	
$\boldsymbol{\Sigma_\nu^{-1}}$	t…/noiseMat_Vec/Sigma_nu_inv_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\boldsymbol{\Sigma_\kappa^{-1}}$	signal/Sigma_kappa_inv_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\boldsymbol{\Sigma_\nu^{-1}\mathbf{W}}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC

R6

$\boldsymbol{\Sigma_\kappa}$	g…/signal/Sigma_kappa_R6_0p02_0p22_bp62_fwriteC
clean/Wspace.m ~ 2.25 days for $\boldsymbol{\Sigma_\nu^{-1}}$, 9.8 days for $\boldsymbol{\Sigma_\kappa^{-1}}$, 6.7 days for diagonalization, 3.4 hours for $\boldsymbol{\Sigma_\nu^{-1}\mathbf{W}}$	
$\boldsymbol{\Sigma_\nu^{-1}}$	t…/noiseMat_Vec/Sigma_nu_inv_[σ_m]_R6_0p02_0p22_bp62_fwriteC
$\boldsymbol{\Sigma_\kappa^{-1}}$	signal/Sigma_kappa_inv_R6_0p02_0p22_bp62_fwriteC
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R6_0p02_0p22_bp62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R6_0p02_0p22_bp62_Values_fwriteR
$\boldsymbol{\Sigma_\nu^{-1}\mathbf{W}}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R6_0p02_0p22_bp62_fwriteC





Because W-space is dominated by shot noise, we expect eigenvectors to be highly localized in cells. The zero-point noise, which is of a lower magnitude, should introduce some leakage of those modes into cells that lie within the same SEGMENT.

Upon inspection, we discovered that for most of the lower order ($\lesssim$ 10,000) modes the eigenvector elements were effectively zero everywhere except in about 1 to 4 adjacent cells, matching our expectations. (The pixels are too small and dim to be visible in this document, so refer to the saved images in the notes folder if you want to examine them.) The eigenvectors become much more diffuse at higher orders.

The test vectors can be rotated into W-space.

$\boldsymbol{\delta}$	`clean/shot/delta_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\boldsymbol{\delta}_\kappa$	`clean/shot/delta_kappa_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\boldsymbol{\delta}_\eta$	`clean/shot/delta_eta_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\mathbf{W}$	`clean/Eigen_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Vectors_fwriteC`
clean/ScriptdRealz.m (E) ~ < 1-6 min for 200 realizations	

$\boldsymbol{d}$	clean/shot/W_d_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{d_s}$	clean/shot/W_ds_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{d_n}$	clean/shot/W_dn_R[r]_[σ_m]_[N_{realz}]_fwriteC

Our cleansing algorithm is underestimating the true signal and overestimating the true noise. This suggests there's some sort of overweighting of the noise in W-space. To quantify this, I modify $\boldsymbol{\Sigma}_\kappa^{-1} + \alpha\boldsymbol{\Sigma}_\nu^{-1} = \mathbf{W}\boldsymbol{\Lambda}^{(\kappa\nu)}\mathbf{W}^T$ by introducing a scalar factor α where $\alpha = \left\{\frac{1}{2}, 2\right\}$. In the interest of time, I only test for R16. This code runs the same as before, but just run the lines with the comment "only use with alpha scaling" instead.

clean/Wspace.m ~ 11hr for diagonalization, 15min for $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$ *(R16)*	
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_alpha[α]_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_alpha[α]_Values_fwriteR
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_alpha[α]_fwriteC

Sampling Algorithm

The trickiest part of running the sampler is calculating α. The independent candidate density $q(\boldsymbol{\theta}) \approx 10^9$ if one ignores the exponent. When taken to the power $-(N+f)/2$, the value reduces to 0. As a result, we need to be clever during the calculation to avoid numerical overruns.

$$\frac{g(\boldsymbol{\theta}'|\boldsymbol{\delta})q(\boldsymbol{\theta})}{g(\boldsymbol{\theta}|\boldsymbol{\delta})q(\boldsymbol{\theta}')} = \frac{\exp\left(\ln\left(g(\boldsymbol{\theta}'|\boldsymbol{\delta})\right)\right)}{\exp\left(\ln\left(g(\boldsymbol{\theta}|\boldsymbol{\delta})\right)\right)} \frac{\exp\left(\ln\left(q(\boldsymbol{\theta})\right)\right)}{\exp\left(\ln\left(q(\boldsymbol{\theta}')\right)\right)}$$

$$= \exp\left(l(\boldsymbol{\theta}'|\boldsymbol{\delta}) - l(\boldsymbol{\theta}|\boldsymbol{\delta}) + \ln\left(a(\boldsymbol{\theta})^{-(N+f)/2}\right) - \ln\left(a(\boldsymbol{\theta}')^{-(N+f)/2}\right)\right)$$

$$= \exp\left(l(\boldsymbol{\theta}'|\boldsymbol{\delta}) - l(\boldsymbol{\theta}|\boldsymbol{\delta}) + \left(\frac{N+f}{2}\right)\left(\ln\left(a(\boldsymbol{\theta}')\right) - \ln\left(a(\boldsymbol{\theta})\right)\right)\right),$$

where

$$a(\boldsymbol{\theta}) \equiv 1 + \left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right)^T \boldsymbol{\Lambda}^{(\kappa\nu)}\left(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}\right).$$

Next, the independent candidate density's degrees of freedom parameter f must be adjusted to admit the optimal number of variates. As f increases, $q(\boldsymbol{\theta})$ approaches a multivariate Gaussian. The ideal acceptance rate for an N-dimensional Gaussian is about 23% when N samples are drawn. For R8, I ran a test where the number of samples drawn from $q(\boldsymbol{\theta})$ is $N = 45{,}506$. I adjusted f each time and observed the following acceptance rate,

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
clean/tDist_fSolver.m ~ 22 min per f (R7)	

Plots.m (P)

The image part with relationship ID rId11 was not found in the file.

An acceptance rate of 23% is reached at $f \cong 104$ for R8, at $f \cong 128$ for R7 and at $f \cong 58$ for R16. Because $\boldsymbol{\delta}$ only shifts the mean of the distribution, the above chart should be independent of the data vector you wish to cleanse.

$\boldsymbol{\delta}$	`clean/shot/delta_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\boldsymbol{\lambda}^{(\kappa\nu)}$	`clean/Eigen_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Values_fwriteR`
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	`t…/clean/Sigma_nu_inv_W_[`σ_m`]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteC`
clean/MetropolisHastings.m ~ 84 min (2hr 53min) to generate 10^5 candidates, 34 min (1hr 11min) to rotate them for R8 (R7)	
$\boldsymbol{\theta}^{(\alpha)}\vert\boldsymbol{\delta}$	`t…/clean/shot/Realz_theta_given_delta[`n`]_R[`r`]_[`σ_m`]_[`N_{realz}`]_[`ID`]_fwriteC`
$\boldsymbol{\kappa}^{(\alpha)}\vert\boldsymbol{\delta}$	`t…/clean/shot/Realz_kappa_given_delta[`n`]_R[`r`]_[`σ_m`]_[`N_{realz}`]_[`ID`]_fwriteC`

Note that in this algorithm the $\boldsymbol{\theta}$ realizations are actually realizations of signal in W-space, i.e. $\left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right) = \left(\boldsymbol{s}^{(\alpha)}|\boldsymbol{\delta}\right)$. It takes roughly half a day to generate a million RVs from $P(\boldsymbol{\theta}|\boldsymbol{\delta})$. Getting the coefficients in cell-space is accomplished through the rotation $\left(\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}\right) = \mathbf{W} * \left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right)$.

Trace Plots

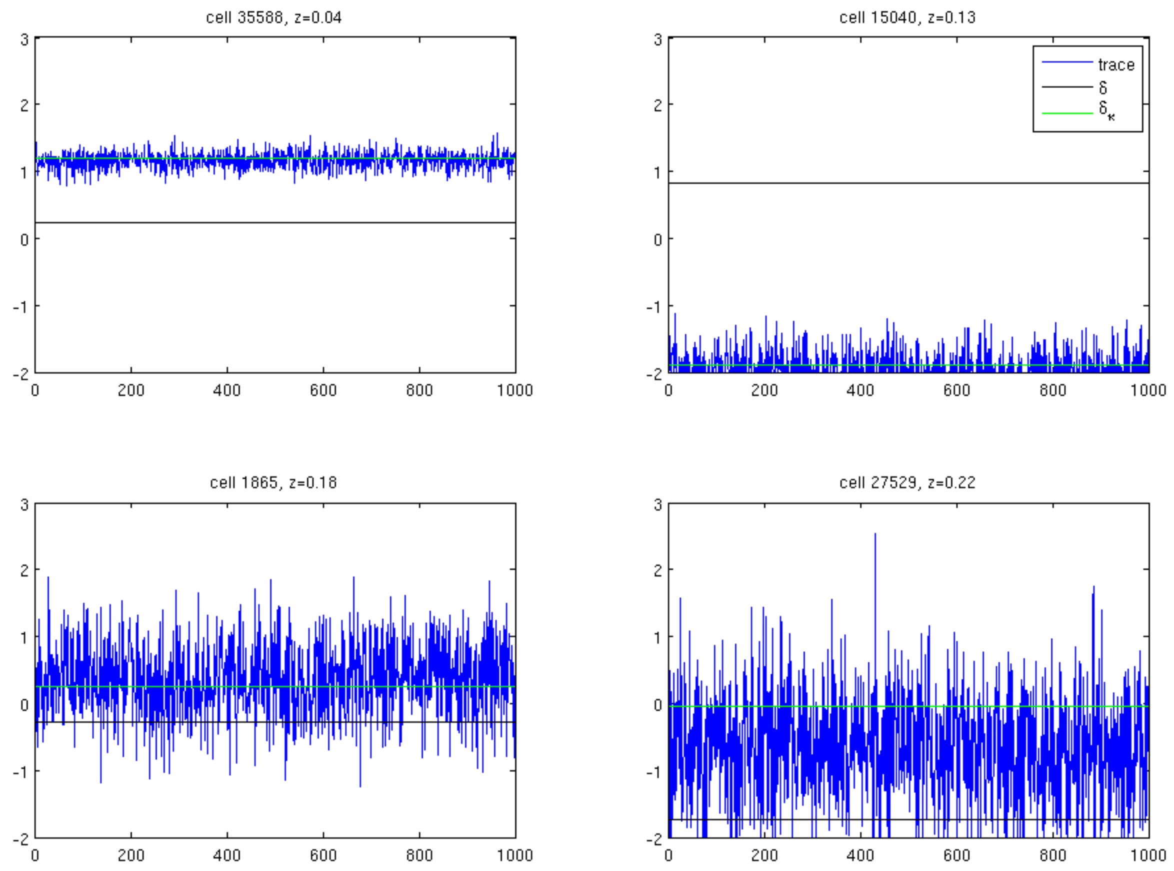
The usual tool for assessing the progress of a Metropolis-Hastings algorithm is known as a "trace plot". Each time a new $\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}$ is drawn, a point is laid down indicating the value of the ith parameter during realization α. For clarity, only one dimension's element is visualized at a time.

If the initial vector $\boldsymbol{\theta}^{(0)}$ is selected far from the peak of the distribution $g(\boldsymbol{\theta}|\boldsymbol{\delta})$, one must continue drawing random vectors $\boldsymbol{\theta}'$ until the region containing the higher probabilities is reached. This is known as the "burn-in" period. The number of vectors needed, and ultimately discarded, during the burn-in period varies with distribution and initial position. On a stable trace plot, i.e. one from which we can accept random variates, the vector elements drawn should vary around some fixed horizontal trend line. A burn-in period typically manifests itself on a trace plot as a trend approaching that baseline.

With this problem I know precisely where $g(\boldsymbol{\theta}|\boldsymbol{\delta})$ peaks, so no burn-in period should be necessary. I verify this assumption by examining the trace plots of four elements. These elements were selected to lie at well-separated redshifts. As one can see in the figures below, the trace plots start out stable meaning we can start accepting $\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}$ right away.

$\boldsymbol{\delta}$	`clean/shot/delta_R`[r]`_`[σ_m]`_`[N_{realz}]`_fwriteC`
$\boldsymbol{\delta}_\kappa$	`clean/shot/delta_kappa_R`[r]`_`[σ_m]`_`[N_{realz}]`_fwriteC`
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	`t…/clean/Sigma_nu_inv_W_`[σ_m]`_R`[r]`_`[z_{min}]`_`[z_{max}]`_bp62_fwriteC`
$\boldsymbol{\lambda}^{(\kappa\nu)}$	`clean/Eigen_W_`[σ_m]`_R`[r]`_`[z_{min}]`_`[z_{max}]`_bp62_Values_fwriteR`
$\mathbf{W}$	`clean/Eigen_W_`[σ_m]`_R`[r]`_`[z_{min}]`_`[z_{max}]`_bp62_Vectors_fwriteC`
$\boldsymbol{\kappa}^{(\alpha)}\|\boldsymbol{\delta}$	`t…/clean/shot/Realz_kappa_given_delta`[n]`_R`[r]`_`[σ_m]`_`[N_{realz}]`_`[ID]`_fwriteC`
clean/TracePlots.m ~ 2 min	

The above routine creates trace plots in cell-space. If you wish to avoid plotting anything related to $\hat{\boldsymbol{\theta}}$, ignore input of files 3, 4 and 5 in the above list. Another routine, *TracePlots2.m*, does the same thing in W-space.



The black line in these figures represents the value of the data element δ_i while the blue line represents the value of the clustering signal component $\delta_{\kappa,i}$. Note that the variates $\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}$ tend to cluster more around the signal than the data even though those variates were generated through $\boldsymbol{\delta}$ *without explicit knowledge of* $\boldsymbol{\delta}_\kappa$. This is exactly the effect we were searching for.

Of course, not every cell has Metropolis-Hastings variates that are right on target with the signal. This probably occurs because those cells are more highly contaminated than others. Can we show this?

Expected Signal and Variance

We solve for the expected signal in each dimension by averaging over all realizations,

$$\hat{\kappa}_i = \frac{1}{K}\sum_{\alpha=1}^{K}\left(\kappa_i^{(\alpha)}|\boldsymbol{\delta}\right).$$

The variance of the mean for uncorrelated random variables is

$$\mathrm{Var}(\hat{\kappa}_i) = \frac{\sigma_{\kappa_i}^2}{K},$$

where $\mathrm{Var}(\kappa_i) = \sigma_{\kappa_i}^2$. This is merely an approximation in this case since random variates drawn through Metropolis-Hastings are technically not independent of one another. However, these variates are drawn from an independent candidate density that blankets the entire distribution $g(\boldsymbol{\theta}|\boldsymbol{\delta})$. This means that essentially all of the parameter space is accessible on each draw and the correlations aren't as strong as they might be otherwise. I might try providing an exact answer, but finding $\mathrm{Cov}\left(\kappa_i^{(\alpha)},\kappa_i^{(\beta)}\right)$ is a more difficult problem than I am willing to undertake at the moment.

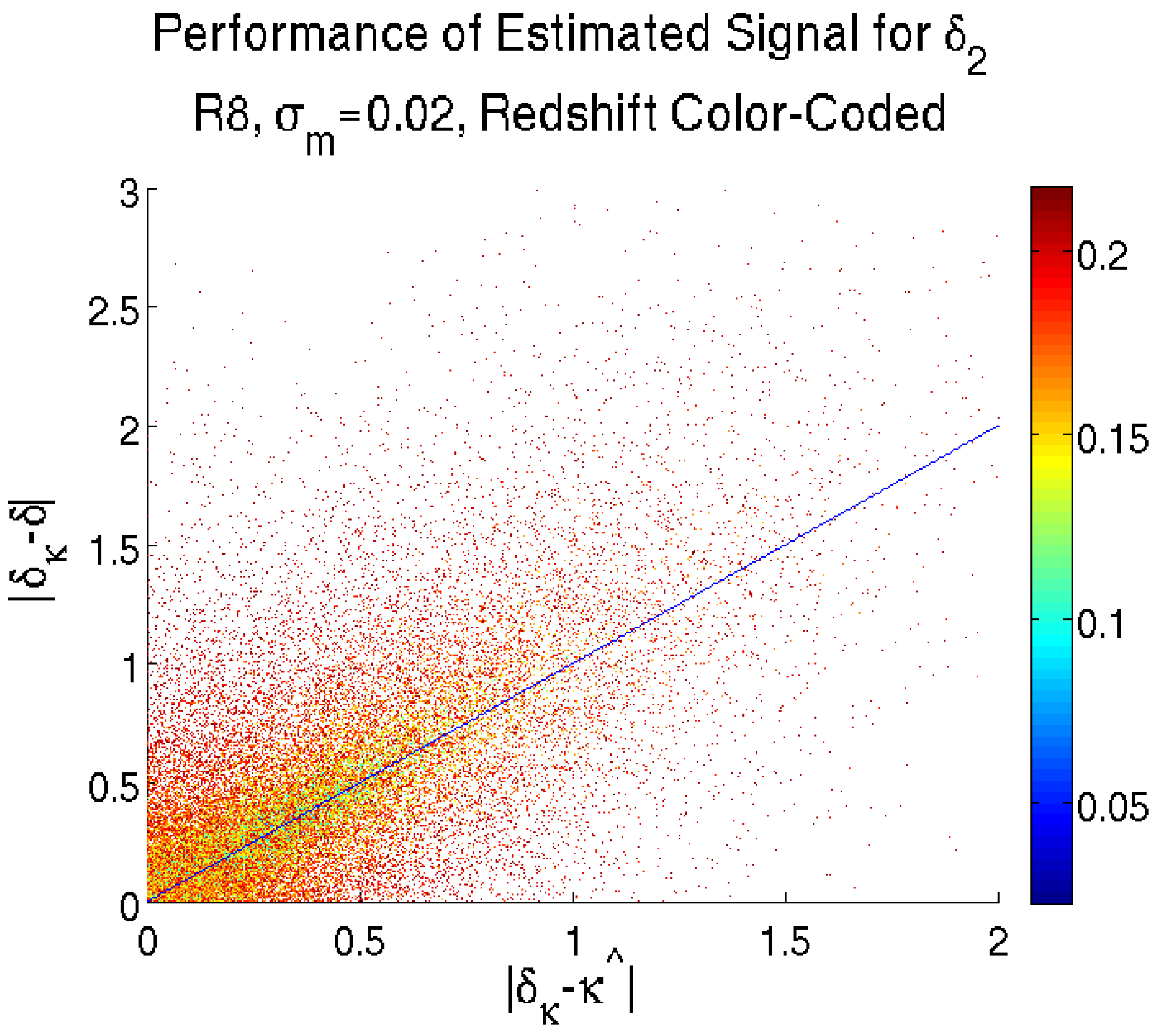
Results – R8

$\sigma_m = 0.02$

To find the mean and variance of the estimated signal, I must consider all $\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}$ files. There isn't enough room in memory to read them all in simultaneously, so I have to do it piecewise.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}$	t…/clean/shot/Realz_kappa_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_[ID]_fwriteC
clean/kappahatAnalysis.m ~ for each 10^5 realizations, 6.7min (10.7min) for the sum, 7min (10.8min) for the variance, R8 (R7)	
$\hat{\kappa}_i, \sigma_{\kappa_i}$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\hat{\kappa}_{K,i}$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_logspace_[a]_[b]_[nl]_fwriteC

The figure below attempts to visualize the distance between the true signal $\boldsymbol{\delta}_\kappa$ and the raw data and estimated signal $\boldsymbol{\delta}$ and $\hat{\kappa}_i$, respectively. For about 58.3% of cells, the true signal $\delta_{\kappa,i}$ lies closer to the estimate $\hat{\kappa}_i$ than to the raw data δ_i. In other words, if given the choice to measure the true clustering signal with $\hat{\boldsymbol{\kappa}}$ or $\boldsymbol{\delta}$, the estimated signal will yield superior performance.



I can also measure the difference in aggregate using the vector 2-norm. Here are the results for simulated data vector 2,

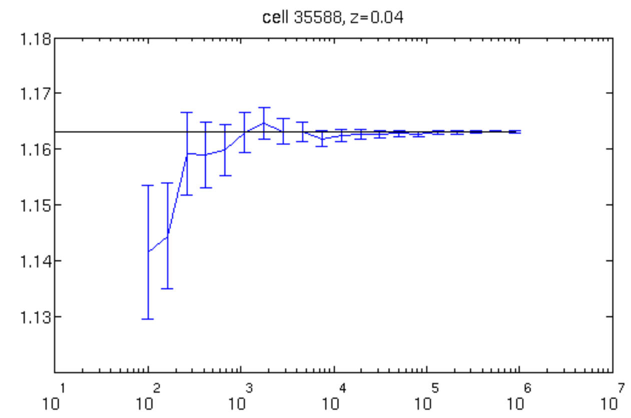
$$\begin{aligned}\|\boldsymbol{\delta}_\kappa-\boldsymbol{\delta}\|_2 &= 131.3708,\\ \|\boldsymbol{\delta}_\kappa-\hat{\boldsymbol{\kappa}}\|_2 &= 103.0903,\\ \|\boldsymbol{\delta}_\zeta\|_2 &= 131.3247.\end{aligned}$$

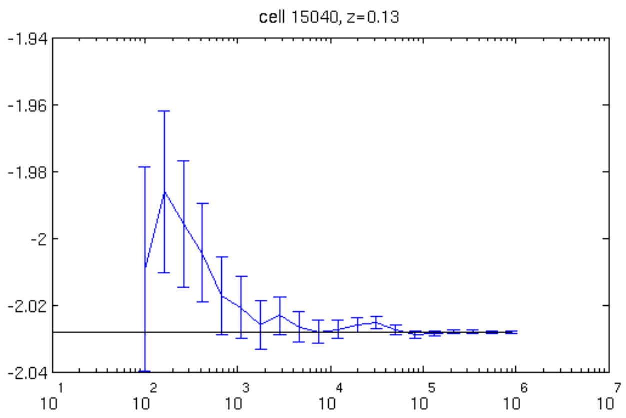
Notice that the estimated signal lies closer to the truth than does the raw data.

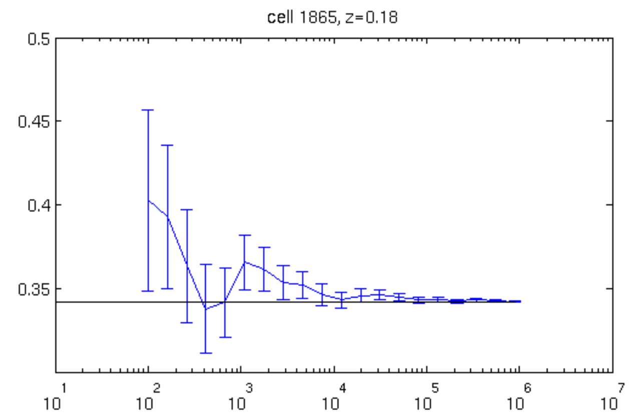
The deviation has decreased by a factor of 131/103 which means the variance with the correction is ~1.6 times, or 38%, lower than the variance without it.

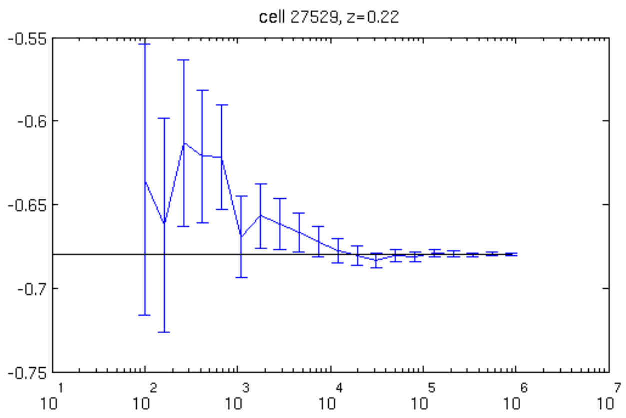
I can also report how the estimated signal converges over time. We can use this to help determine the number of realizations needed to achieve an accuracy of acceptable accuracy and precision.

Recalculating $\sigma_{\kappa_i}^2$ for each K is computationally expensive and it doesn't change much as a function of the number of samples anyway. Therefore, I only use $\sigma_{\kappa_i}^2$ calculated from the totality of the samples when plotting the error bars below.





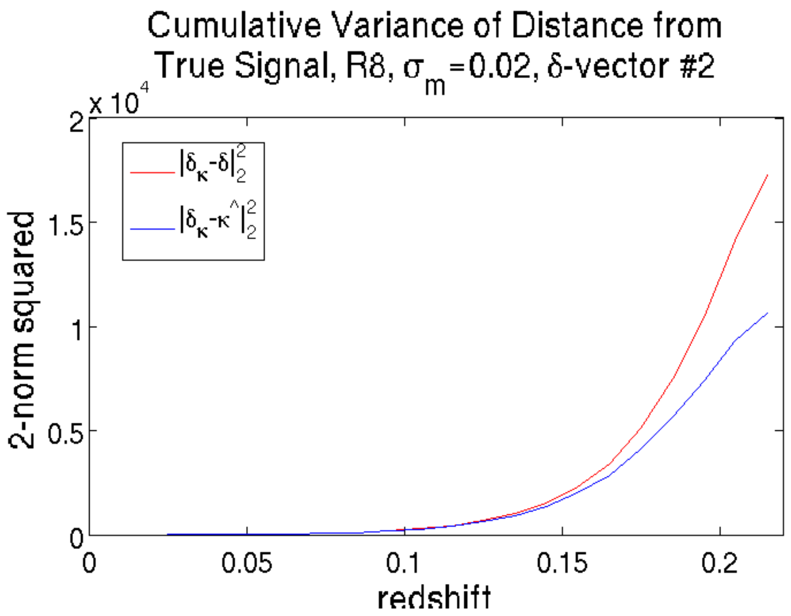




The horizontal axis conveys the number of realizations K that went into the signal estimate. The error bars are the 1 σ uncertainty in the mean $\hat{\kappa}_i$. This plot was created for test data vector 1, R8 with $\sigma_m = 0.02$.

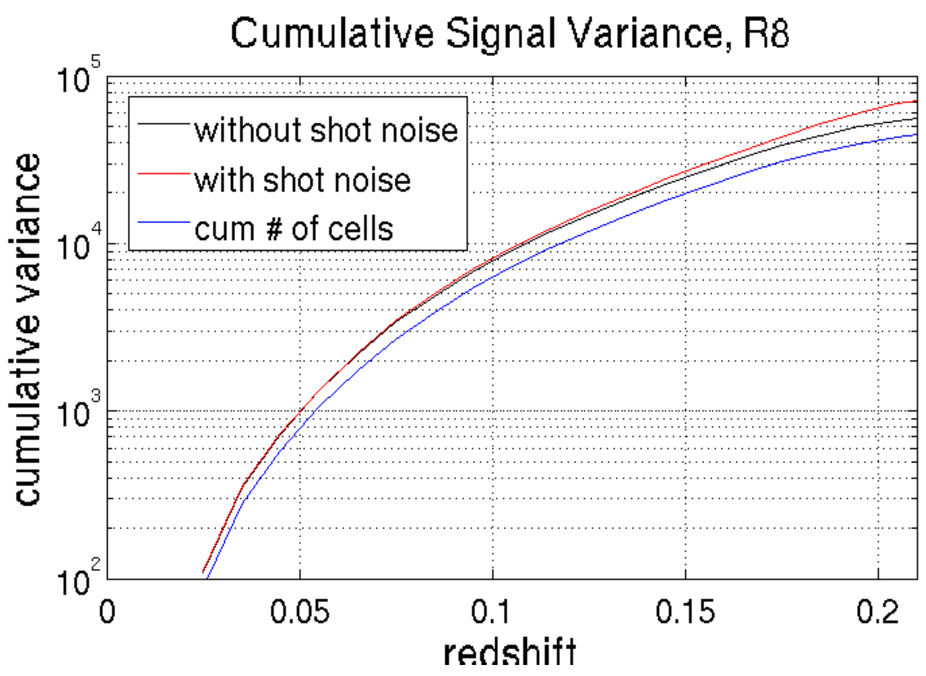
There is a clear redshift dependence in the cleansing. The distance between the raw data and true underlying signal divergences as more high-redshift cells are included. When the raw data is replaced with the signal estimate $\hat{\boldsymbol{\kappa}}$, the distance still increases but more slowly than the alternative.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\hat{\kappa}_i$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_fwriteC
clean/kappahatz.m	

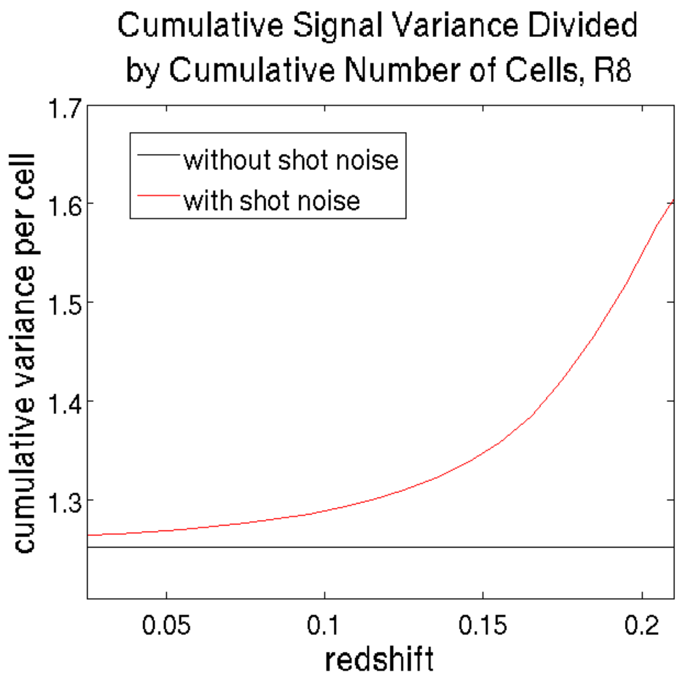


For comparison, here's the same plot but for the cumulative variance of the clustering signal with and without shot noise. The variances are gathered from the diagonal elements of $\mathbf{\Sigma}_\kappa$.

$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \lvert\mathbf{d}\rvert, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n \rangle, \delta$	`base/Overdensities_DR6_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteR`
$\mathbf{\Sigma}_{\boldsymbol{\kappa}}$	`signal/Sigma_kappa_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteC`
SmallCode.m (I) ~ fast	



Can we use a plot like this to know where to cut our cells off – a point where cutting would give us a better signal-to-noise? Are the most distant cells contributing much to the cumulative signal? Do they contain a lot of real information? If the shot noise increases faster than the cosmological signal but slower than the number of cells, then we might still be ahead.



$\boldsymbol{\sigma_m = 0.24}$

The image part with relationship ID rId18 was not found in the file.

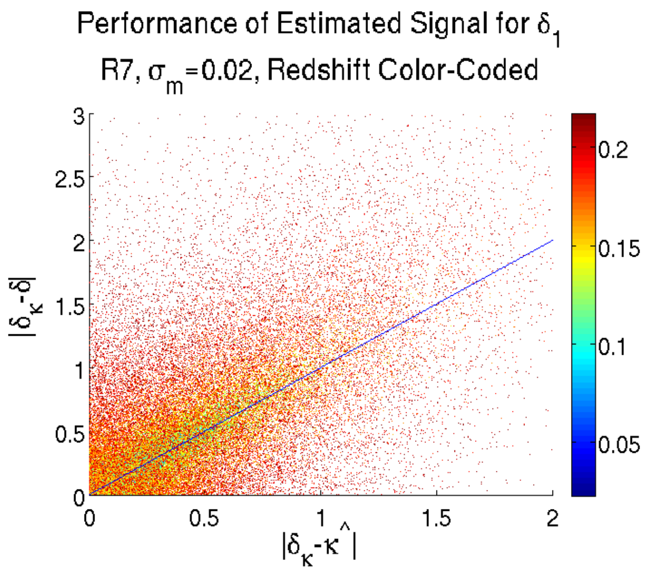
I can also measure the difference in aggregate using the vector 2-norm. Here are the results for first 50 $\hat{\boldsymbol{\kappa}}$'s.

$$\|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2 = 135.8 \pm 0.8$$
$$\|\boldsymbol{\delta}_\kappa - \hat{\boldsymbol{\kappa}}\|_2 = 104.3 \pm 0.5$$

Notice that the estimated signal lies closer to the truth than does the raw data. The variance with the correction is $\sim$1.7 times, or 41%, lower than the variance without it.

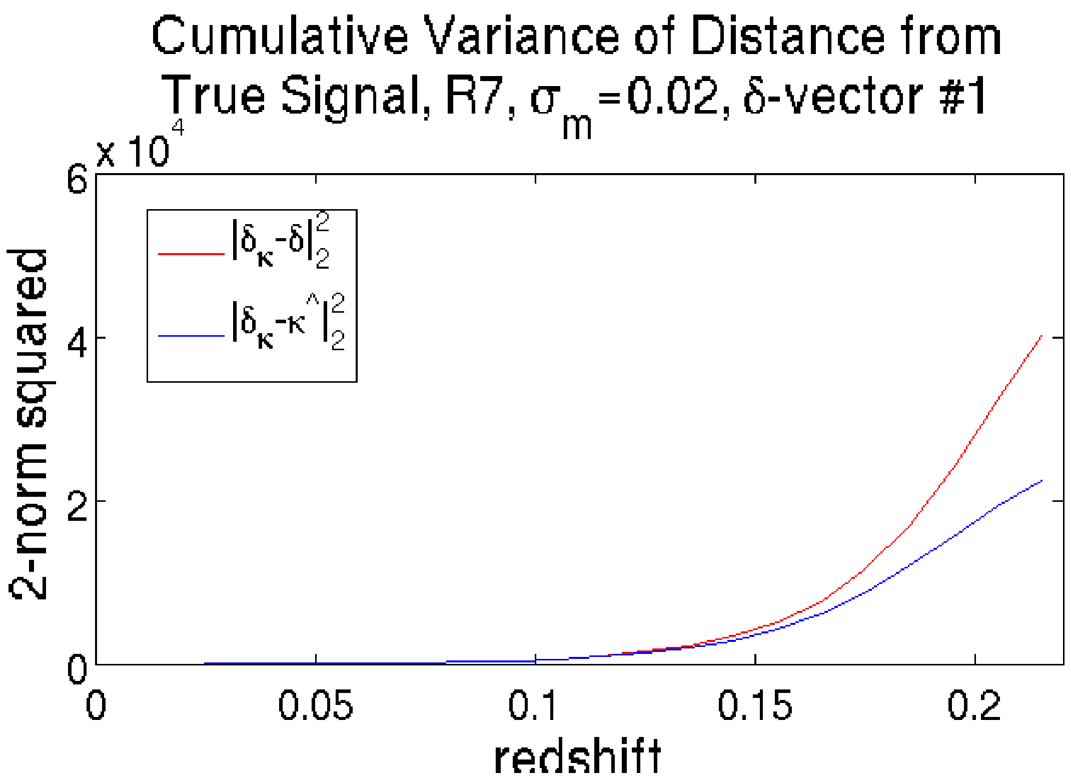
Results – R7

The percentage of cells that saw improvement increased to 59.7% for R7 from 58.3% for R8.

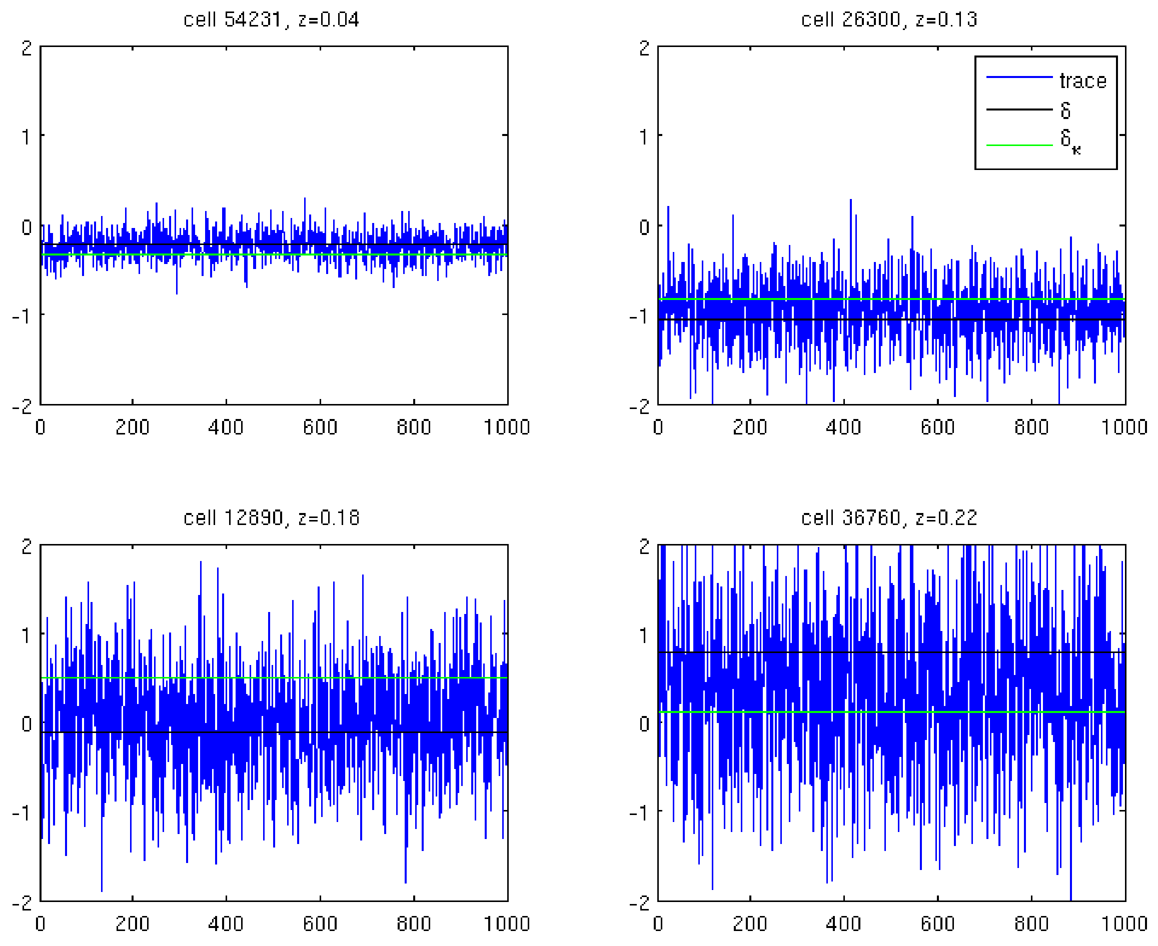


$$\begin{aligned}\|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2 &= 200.3153,\\ \|\boldsymbol{\delta}_\kappa - \widehat{\boldsymbol{\kappa}}\|_2 &= 149.4259,\\ \|\boldsymbol{\delta}_\zeta\|_2 &= 200.2437.\end{aligned}$$

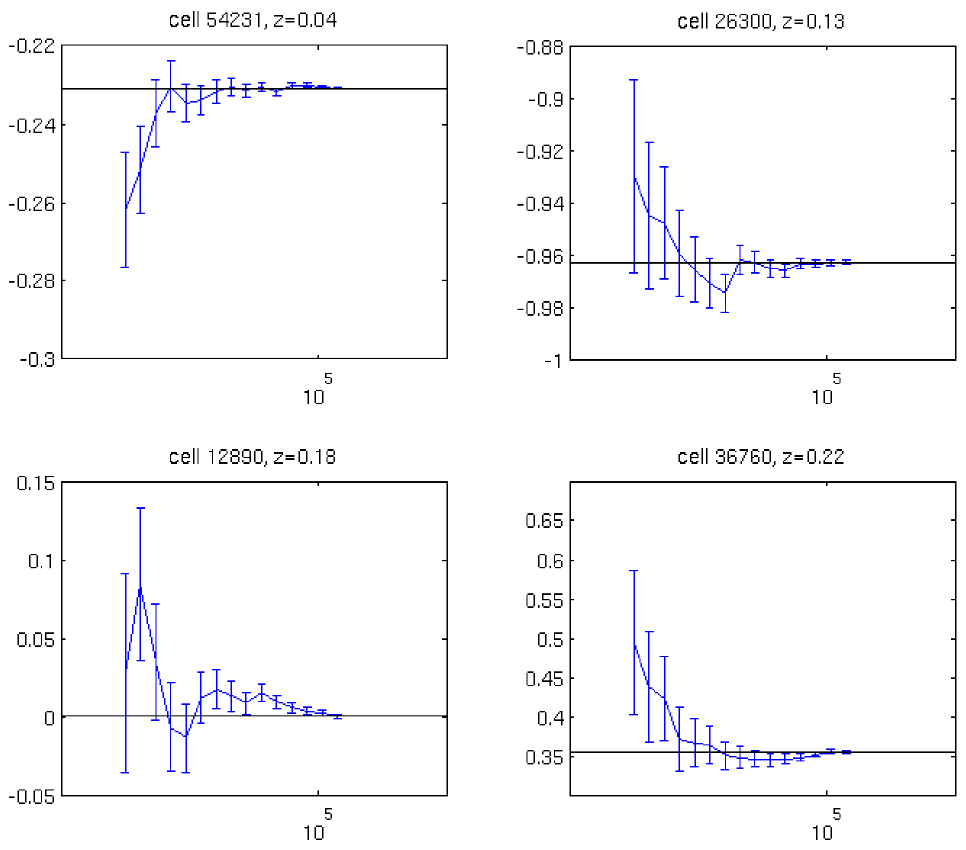
Notice that the estimated signal lies closer to the truth than does the raw data. The deviation has decreased by a factor of 200/149 which means the variance with the correction is ~1.8 times, or 44%, lower than the variance without it.



Here are some trace plots.

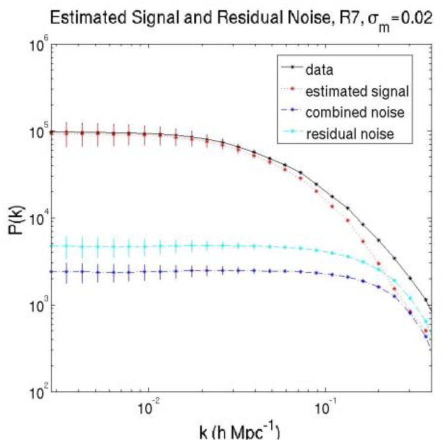


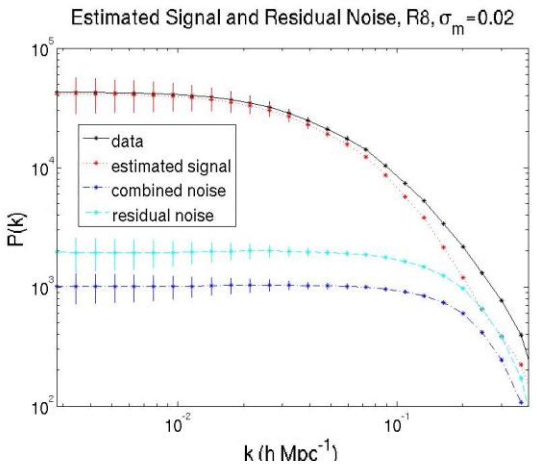
The convergence properties for R7 are similar to those for R8.



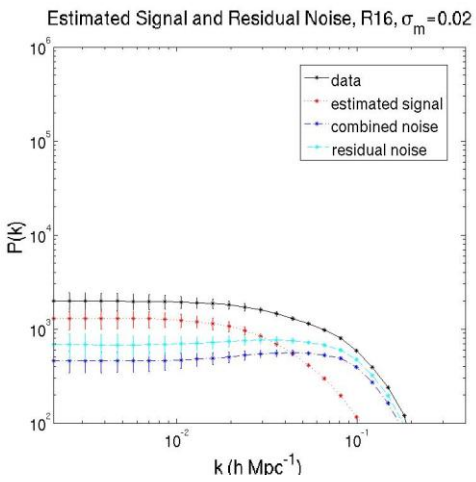
Residual Noise

Under ideal circumstances the power of the deprojected noise, $\boldsymbol{\delta} - \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$, would exactly equal the sum of the power of the zero-point and shot noise terms. As shown above, our method removes a bit more noise than is actually present. This leads to $\langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$ somewhat underestimating $\boldsymbol{\kappa}$ and the residual $\boldsymbol{\delta} - \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$ somewhat overestimating the actual amount of noise present $\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta$.





[figure: power spectrum of residual noise between the raw data and estimated signal is shown relative to true noise (i.e. shot noise plus zero-point noise). $\sigma_m = 0.02$.]



Faster Signal Estimates

While placing error bars on $\hat{\boldsymbol{\kappa}}$ requires knowing the full set of $\boldsymbol{\kappa}^{(\alpha)}$ realizations, finding the value of $\hat{\boldsymbol{\kappa}}$ itself does not. If speed is a concern, one can avoid the rotations $\left(\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}\right) = \mathbf{W} * \left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right)$ performed for each batch of realizations in W-space and instead do one rotation of the averaged $\left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right)$ at the very end.

In other words, here's how the signal estimated is calculated if the random variates have already been rotated back into cell-space,

$$\hat{\kappa}_i = \frac{1}{K}\sum_{\alpha=1}^{K}\left(\kappa_i^{(\alpha)}|\boldsymbol{\delta}\right).$$

If the variables are still in W-space,

$$\hat{\kappa}_i = \frac{1}{K}\sum_{\alpha=1}^{K}\mathbf{W} * \left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right)$$

$$= \mathbf{W} * \left(\frac{1}{K}\sum_{\alpha=1}^{K}\left(\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}\right)\right).$$

I constructed a single efficient routine that both generates the Metropolis-Hastings RVs and aggregates them to solve for $\hat{\boldsymbol{\kappa}}$ without saving any realizations to disk. After the initialization portion of the code, the user can specify the data vectors for which he wishes to estimate the signal at the line `dd=40:43`. In this case the code would solve for $\hat{\boldsymbol{\kappa}}$ for $\boldsymbol{\delta}$ vectors 40, 41, 42 and 43. Just be sure that there is a file name provided in the initialization for every $\hat{\boldsymbol{\kappa}}$ solution you want to write to disk. If the user runs this on 10 MATLAB windows simultaneously, he can easily generate at least 40 $\hat{\boldsymbol{\kappa}}$ solutions overnight.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
clean/MHkappahat.m ~ 1.8hrs per run at 10^5 *realizations, R8* *3.2hrs per run at* 10^5 *realizations, R7*	
$\hat{\boldsymbol{\kappa}}$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_1e5_fwriteC

At the end of this code I have included some scripts to perform analysis and create plots like the ones presented above.

Before I created the efficient routine, I tested my method by creating two separate pieces of code that handled the Metropolis-Hastings generation and $\hat{\boldsymbol{\kappa}}$ solver in separate steps. It did this by writing giant $\boldsymbol{\theta}^{(\alpha)}|\boldsymbol{\delta}$ files to disk in the first step and opening/processing them in the second. This method allows you to measure the convergence of the estimator, but once that's been figured out once, there's really no need to do it again. For completeness, here are those two routines.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	t…/clean/Sigma_nu_inv_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
clean/MetropolisHastingsFast.m ~ 84min (2hr 43min) to generate 10^5 *candidates, R8 (R7)*	
$\boldsymbol{\theta}^{(\alpha)}\|\boldsymbol{\delta}$	t…/clean/shot/Realz_theta_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_[ID]_fwriteC

Likewise, the $\hat{\boldsymbol{\kappa}}$ solver is modified only slightly by changing the input realization files and introducing the rotation through $\mathbf{W}$.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\mathbf{W}$	clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\theta}^{(\alpha)}\|\boldsymbol{\delta}$	t…/clean/shot/Realz_theta_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_[ID]_fwriteC
clean/kappahatAnalysisFast.m ~ 10.2 min/1e5 realz R7	
$\hat{\kappa}_i$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\hat{\kappa}_{K,i}$	clean/shot/kappahatK_given_delta[n]_R[r]_[σ_m]_logspace_[a]_[b]_[nl]_fwriteC

Expected Shot Noise and Zero-Point Noise

We assume the shot noise $\boldsymbol{\zeta}(\boldsymbol{\varphi})$ is mean-zero and Gaussian with the distribution function,

$$P(\boldsymbol{\varphi}) \propto \exp\left[-\frac{1}{2}\boldsymbol{\zeta}(\boldsymbol{\varphi})^T \boldsymbol{\Sigma}_\zeta^{-1} \boldsymbol{\zeta}(\boldsymbol{\varphi})\right].$$

The signal plus systematic noise is also Gaussian, therefore $\boldsymbol{\delta}$ is symmetric around $\boldsymbol{\zeta}$ such that the probability of obtaining a particular data vector given a set of shot noise parameters $\boldsymbol{\varphi}$ is

$$P(\boldsymbol{\delta}|\boldsymbol{\varphi}) \propto \exp\left[-\frac{1}{2}\big(\boldsymbol{\delta} - \boldsymbol{\zeta}(\boldsymbol{\varphi})\big)^T \boldsymbol{\Sigma}_{\kappa\eta}^{-1} \big(\boldsymbol{\delta} - \boldsymbol{\zeta}(\boldsymbol{\varphi})\big)\right],$$

where $\boldsymbol{\Sigma}_{\kappa\eta}^{-1} = \left(\boldsymbol{\Sigma}_\kappa + \boldsymbol{\Sigma}_\eta\right)^{-1}$. Let us further introduce the eigenbasis $\mathbf{B}$ which results from the following diagonalization,

$$\boldsymbol{\Sigma}_\zeta^{-1} + \boldsymbol{\Sigma}_{\kappa\eta}^{-1} = \mathbf{B}\boldsymbol{\Lambda}^{(B)}\mathbf{B}^T.$$

A data vector represented in B-*space* is denoted

$$\boldsymbol{\xi} = \boldsymbol{\omega} + \boldsymbol{\phi} + \boldsymbol{\pi}, \qquad B\ space$$

where

$$\boldsymbol{\pi} = \mathbf{B}^T\boldsymbol{\zeta}, \qquad \boldsymbol{\zeta} = \mathbf{B}\boldsymbol{\pi}.$$

Taking the product of the probabilities,

$$P(\boldsymbol{\delta}|\boldsymbol{\varphi})P(\boldsymbol{\varphi}) \propto \exp\left[\boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\kappa\eta}^{-1}\boldsymbol{\zeta}(\boldsymbol{\varphi})\right] \exp\left[-\frac{1}{2}\boldsymbol{\zeta}(\boldsymbol{\varphi})^T\left(\boldsymbol{\Sigma}_\zeta^{-1} + \boldsymbol{\Sigma}_{\kappa\eta}^{-1}\right)\boldsymbol{\zeta}(\boldsymbol{\varphi})\right]$$

$$\propto \exp\left[\boldsymbol{\delta}^T \boldsymbol{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}\,\boldsymbol{\pi}(\boldsymbol{\varphi})\right] \exp\left[-\frac{1}{2}\boldsymbol{\pi}(\boldsymbol{\varphi})^T\boldsymbol{\Lambda}^{(B)}\boldsymbol{\pi}(\boldsymbol{\varphi})\right]$$

$$\propto \exp\left[\mathbf{h}^T\boldsymbol{\pi}(\boldsymbol{\varphi}) - \frac{1}{2}\boldsymbol{\pi}(\boldsymbol{\varphi})^T\boldsymbol{\Lambda}^{(B)}\boldsymbol{\pi}(\boldsymbol{\varphi})\right],$$

where $\mathbf{h}^T = \boldsymbol{\delta}^T\boldsymbol{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}$.

The expected value of the i^{th} shot noise coefficient in B-space is calculated through the equation below. We have chosen $\boldsymbol{\varphi} = \boldsymbol{\pi}$ as the shot noise parameters. The second equality comes from Bayes's theorem.

$$\langle \pi_i|\boldsymbol{\delta}\rangle = \int \pi_i\, P(\boldsymbol{\pi}|\boldsymbol{\delta})\, d\boldsymbol{\pi} = \int \pi_i\, P(\boldsymbol{\delta}|\boldsymbol{\pi})P(\boldsymbol{\pi})\, d\boldsymbol{\pi}.$$

Following a similar argument to that employed for W-space, we find

$$\langle \pi_i | \boldsymbol{\delta} \rangle = \frac{h_i}{\lambda_i^{(B)}}.$$

Finally, rotate the result back into cell-space with $\langle \boldsymbol{\zeta} | \boldsymbol{\delta} \rangle = \mathbf{B} \langle \boldsymbol{\pi} | \boldsymbol{\delta} \rangle$.

Efficiently Solving for $\boldsymbol{\Sigma}_{\kappa\eta}^{-1}$

One can evaluate $\boldsymbol{\Sigma}_{\kappa\eta}^{-1} = \left(\boldsymbol{\Sigma}_\kappa + \boldsymbol{\Sigma}_\eta\right)^{-1}$ relatively quickly if $\boldsymbol{\Sigma}_\kappa^{-1}$ and the diagonalization $\boldsymbol{\Sigma}_\eta = \mathbf{U}\boldsymbol{\Lambda}^{(\eta)}\mathbf{U}^T$ are already known.

$$\begin{aligned}\boldsymbol{\Sigma}_{\kappa\eta}^{-1} &= \left(\boldsymbol{\Sigma}_\kappa + \boldsymbol{\Sigma}_\eta\right)^{-1} \\ &= \left(\mathbf{U}(\mathbf{U}^T\boldsymbol{\Sigma}_\kappa\mathbf{U})\mathbf{U}^T + \mathbf{U}\boldsymbol{\Lambda}^{(\eta)}\mathbf{U}^T\right)^{-1} \\ &= \mathbf{U}\left(\mathbf{U}^T\boldsymbol{\Sigma}_\kappa\mathbf{U} + \boldsymbol{\Lambda}^{(\eta)}\right)^{-1}\mathbf{U}^T.\end{aligned}$$

By the Sherman-Morrison-Woodbury formula,

$$(\mathbf{S} + \boldsymbol{\Lambda})^{-1} = \mathbf{S}^{-1} - \mathbf{S}^{-1}(\mathbf{S}^{-1} + \boldsymbol{\Lambda}^{-1})^{-1}\mathbf{S}^{-1}.$$

Let $\mathbf{S} \equiv \mathbf{U}^T\boldsymbol{\Sigma}_\kappa\mathbf{U}$ and $\mathbf{T} \equiv \mathbf{S}^{-1} = \mathbf{U}^T\boldsymbol{\Sigma}_\kappa^{-1}\mathbf{U}$.

$$\mathbf{T} = \begin{bmatrix}\mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22}\end{bmatrix}, \qquad \mathbf{T}_c \equiv \begin{bmatrix}\mathbf{T}_{11} \\ \mathbf{T}_{21}\end{bmatrix}, \qquad \mathbf{T}_r \equiv [\mathbf{T}_{11} \quad \mathbf{T}_{12}],$$

and

$$\boldsymbol{\Lambda} = \begin{bmatrix}\mathbf{D} & 0 \\ 0 & 0\end{bmatrix},$$

where $\mathbf{T}_{11}, \mathbf{D} \in \mathbb{R}^{m\times m}$, $\mathbf{T}_{12} \in \mathbb{R}^{m\times(n-m)}$, and $\mathbf{T}_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$. Then,

$$\begin{aligned}(\mathbf{S} + \boldsymbol{\Lambda})^{-1} &= \mathbf{T} - \mathbf{T}(\mathbf{T} + \boldsymbol{\Lambda}^{-1})^{-1}\mathbf{T} \\ &= \mathbf{T} - \mathbf{T}(\mathbf{T} + \boldsymbol{\Lambda}^{-1})^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{\Lambda}\mathbf{T} \\ &= \mathbf{T} - \mathbf{T}(\boldsymbol{\Lambda}\mathbf{T} + \mathbf{I})^{-1}\boldsymbol{\Lambda}\mathbf{T}\end{aligned}$$

Simplify the matrix inverse by partitioning it, then use the formula for the inverse of a partitioned matrix,

$$(\boldsymbol{\Lambda}\mathbf{T} + \mathbf{I})^{-1} = \begin{bmatrix}\mathbf{I} + \mathbf{D}\mathbf{T}_{11} & \mathbf{D}\mathbf{T}_{12} \\ 0 & \mathbf{I}\end{bmatrix}^{-1} = \begin{bmatrix}(\mathbf{I} + \mathbf{D}\mathbf{T}_{11})^{-1} & -(\mathbf{I} + \mathbf{D}\mathbf{T}_{11})^{-1}\mathbf{D}\mathbf{T}_{12} \\ 0 & \mathbf{I}\end{bmatrix}.$$

Completing the multiplication, $(\mathbf{S}+\mathbf{\Lambda})^{-1} = \mathbf{T} - \mathbf{T}_c(\mathbf{I}+\mathbf{D}\mathbf{T}_{11})^{-1}\mathbf{D}\mathbf{T}_r$. Typically the eigenvalues stored in $\mathbf{\Lambda}^{(\eta)}$ are calculated for $\sigma_m = 1$. To adjust for an arbitrary scaling of the photometric zero-points,

$$\left(\mathbf{U}^T\mathbf{\Sigma}_\kappa\mathbf{U} + \mathbf{\Lambda}^{(\eta)}\right)^{-1} = \mathbf{T} - \sigma_m^2\mathbf{T}_c(\mathbf{I}+\sigma_m^2\mathbf{D}\mathbf{T}_{11})^{-1}\mathbf{D}\mathbf{T}_r.$$

I tested this algorithm explicitly for R8 by multiplying $\mathbf{\Sigma}_{\kappa\eta}^{-1}$ from the above equation with $\mathbf{\Sigma}_\kappa + \mathbf{\Sigma}_\eta$ and verifying that the result was the identity matrix.

$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \lvert\boldsymbol{d}\rvert, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n\rangle, \delta$	base/Overdensities_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR
$\mathbf{\Sigma}_\kappa^{-1}$	signal/Sigma_kappa_inv_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{U}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\eta)}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Values.txt
clean/Bspace.m ~ (1hr) for $\mathbf{\Sigma}_{\kappa\eta}^{-1}$ *(R8), (3.6 days) for* $\boldsymbol{B}$ *(R8), (15min) for* $\mathbf{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}$ *(R8)*	
$\mathbf{\Sigma}_{\kappa\eta}^{-1}$	g…/clean/Sigma_KappaEta_inv_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}$	g…/clean/Sigma_KappaEta_inv_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{B}$	clean/Eigen_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(B)}$	clean/Eigen_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR

Testing the Shot Noise Estimator

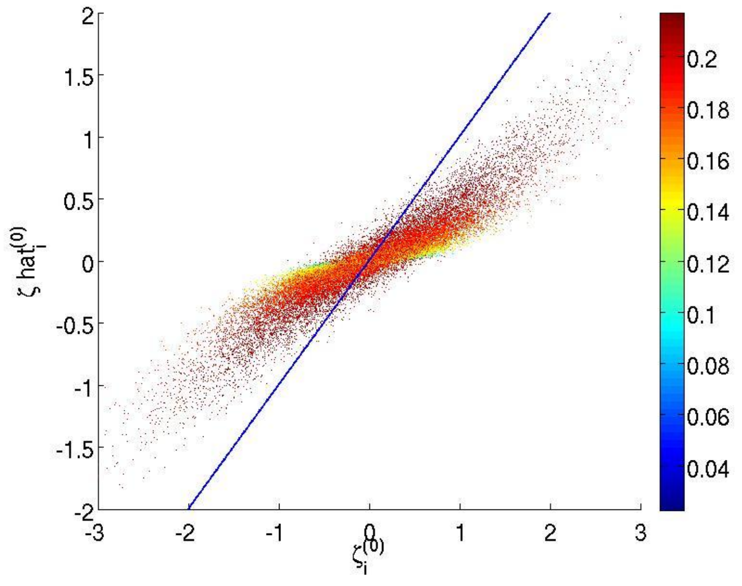
I assess the quality of the shot noise estimator in two ways. First, I use one fixed shot noise overdensity vector $\boldsymbol{\zeta}^{(0)}$ with K signal and zero-point realizations such that $\delta^{(\alpha)} = \kappa^{(\alpha)} + \eta^{(\alpha)} + \zeta^{(0)}$. I estimate the shot noise $\langle\boldsymbol{\zeta}^{(\alpha)}|\boldsymbol{\delta}\rangle$ for each and average the estimates for each cell,
3

$$\hat{\zeta}_i^{(\mathbf{0})} = \frac{\sum_{\alpha=1}^{K}\langle\zeta_i^{(\alpha)}|\boldsymbol{\delta}\rangle}{K}.$$

Second, I use multiple realizations of $\zeta^{(\alpha)}$ and predict $\langle\boldsymbol{\zeta}^{(\alpha)}|\boldsymbol{\delta}\rangle$ for each $\delta^{(\alpha)} = \kappa^{(\alpha)} + \eta^{(\alpha)} + \zeta^{(\alpha)}$. I assume a default guess for the shot noise in each cell is 0 and define the *default error* as $\left|\zeta_i^{(\alpha)}\right|$. I compare this against the *estimate error* $\left|\zeta_i^{(\alpha)} - \langle\zeta_i^{(\alpha)}|\boldsymbol{\delta}\rangle\right|$. For each cell, I average each set of errors and see whether the estimate is closer to the true shot noise than zero.

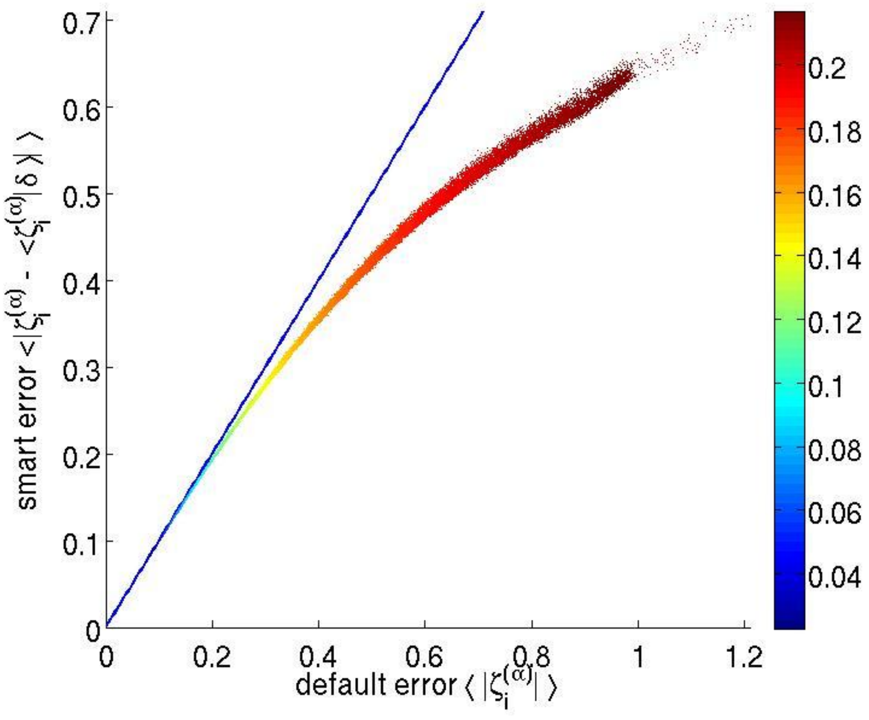
$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \lvert\boldsymbol{d}\rvert, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n\rangle, \delta$	`base/Overdensities_DR6_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteR`
$\mathbf{Z}$	`signal/Eigen_kappa_R[`r`]_[`z_{min}`]_[`z_{max}`]_beta_0p62_Vectors_fwriteC`
$\boldsymbol{\lambda}^{(\kappa)}$	`signal/Eigen_kappa_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Values_fwriteR`
$\mathbf{A}$	`cells_grids/A_DR6_R[`r`]_[ `z_{min}`]_[`z_{max}`]_bp62_fwriteR`
$\boldsymbol{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}$	`g…/clean/Sigma_KappaEta_inv_B_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_fwriteC`
$\mathbf{B}$	`clean/Eigen_B_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Vectors_fwriteC`
$\boldsymbol{\lambda}^{(B)}$	`clean/Eigen_B_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Values_fwriteR`
clean/zgd.m ~ 4 to 15 min per section	
$\zeta_i^{(0)}, \hat{\zeta}_i^{(0)}, z$	`clean/ZetaTestSingle_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_[`nr`].txt`
$\langle\lvert\zeta_i^{(\alpha)}\rvert\rangle, \langle\lvert\zeta_i^{(\alpha)} - \langle\zeta_i^{(\alpha)}\lvert\boldsymbol{\delta}\rangle\rvert\rangle, z$	`clean/ZetaTestMultiple_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_[`nr`].txt`

Plots are created in `paper1.m`. This first plot shows the values in each cell of the simulated shot noise $\boldsymbol{\zeta}^{(0)}$ and the average of the predicted shot noise $\hat{\boldsymbol{\zeta}}^{(0)}$ using multiple signal and systematic noise realizations.



My algorithm does a significantly better job of estimating the shot noise than a random guess would. It is mostly able to pick out the sign of the overdensity and there is a positive correlation between the magnitude of $\zeta_i^{(0)}$ and $\hat{\zeta}_i^{(0)}$. As expected, the shot noise of low redshift objects is less than that of high redshift objects. The values along the y-axis are the result of averaging over 10,000 realizations.

The next plot shows the expected default error and smart error using 10,000 unique realizations of signal, shot noise and systematic noise. This figure shows that at low redshifts there is very little difference between assuming a default shot noise of 0 and using my estimate. For higher redshift cells where the shot noise is larger, using my estimate is preferable to assuming the shot noise is zero.



Testing the Zero-Point Noise Estimator

Because $\mathbf{\Sigma}_\eta = \mathbf{U}\mathbf{\Lambda}^{(\eta)}\mathbf{U}^T$ has a rank $m < N$, $N - m$ diagonal elements of $\mathbf{\Lambda}^{(\eta)}$ equal zero. This makes it impossible to directly evaluate $\mathbf{\Sigma}_\eta^{-1} = \mathbf{U}\mathbf{\Lambda}^{(\eta),-1}\mathbf{U}^T$ since $N - m$ of the diagonal elements of $\mathbf{\Lambda}^{(\eta),-1}$ equal infinity. Therefore when solving for $\langle \boldsymbol{\eta} | \boldsymbol{\delta} \rangle$ we must employ an approach that sidesteps the need for $\mathbf{\Sigma}_\eta^{-1}$.

I suggest the following. Because $\boldsymbol{\delta} = \boldsymbol{\kappa} + \boldsymbol{\eta} + \boldsymbol{\zeta}$,

$$\langle \boldsymbol{\delta} | \boldsymbol{\delta} \rangle = \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle + \langle \boldsymbol{\eta} | \boldsymbol{\delta} \rangle + \langle \boldsymbol{\zeta} | \boldsymbol{\delta} \rangle,$$

and thus,

$$\langle \boldsymbol{\eta} | \boldsymbol{\delta} \rangle = \boldsymbol{\delta} - \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle - \langle \boldsymbol{\zeta} | \boldsymbol{\delta} \rangle.$$

I assess the quality of this estimator in several ways. First, I generate a single zero-point realization $\Delta\boldsymbol{m}^{(0)}$ and an associated $\boldsymbol{\eta}^{(0)} = \mathbf{A}\cdot\Delta\boldsymbol{m}^{(0)}$. I combine this into K data realizations with randomized signal and shot noise vectors $\boldsymbol{\delta}^{(\alpha)} = \boldsymbol{\kappa}^{(\alpha)} + \boldsymbol{\eta}^{(0)} + \boldsymbol{\zeta}^{(\alpha)}$. Each realization will admit a solution for the expected signal and shot noise, and consequently for the zero-point noise as well,

$$\langle\boldsymbol{\eta}^{(\alpha)}|\boldsymbol{\delta}\rangle = \boldsymbol{\delta} - \langle\boldsymbol{\kappa}^{(\alpha)}|\boldsymbol{\delta}\rangle - \langle\boldsymbol{\zeta}^{(\alpha)}|\boldsymbol{\delta}\rangle.$$

The zero-point solutions can be averaged over to provide a best estimate in each cell,

$$\hat{\eta}_{\boldsymbol{i}}^{(0)} = \frac{\sum_{\alpha=1}^{K}\langle\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}|\boldsymbol{\delta}\rangle}{K}.$$

Second, we use $\hat{\boldsymbol{\eta}}^{(0)}$ to compute a best-fit set of photometric coefficients $\Delta\hat{\boldsymbol{m}}^{(0)}$ by solving $\hat{\boldsymbol{\eta}}^{(0)} = \mathbf{A}\cdot\Delta\hat{\boldsymbol{m}}^{(0)}$. These can be compared against $\Delta\boldsymbol{m}^{(0)}$ and associated with the SEGMENT lengths to see whether longer SEGMENTs are better constrained.

We can also solve for $\langle\Delta\boldsymbol{m}^{(\alpha)}|\boldsymbol{\delta}\rangle$**,** the photometric zero-points per realization via $\langle\boldsymbol{\eta}^{(\alpha)}|\boldsymbol{\delta}\rangle = \mathbf{A}\cdot\langle\Delta\boldsymbol{m}^{(\alpha)}|\boldsymbol{\delta}\rangle$. The average of these individual solutions can be reported,

$$\Delta\widetilde{m}_{\boldsymbol{i}}^{(0)} = \frac{\sum_{\alpha=1}^{K}\langle\Delta m_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}|\boldsymbol{\delta}\rangle}{K}.$$

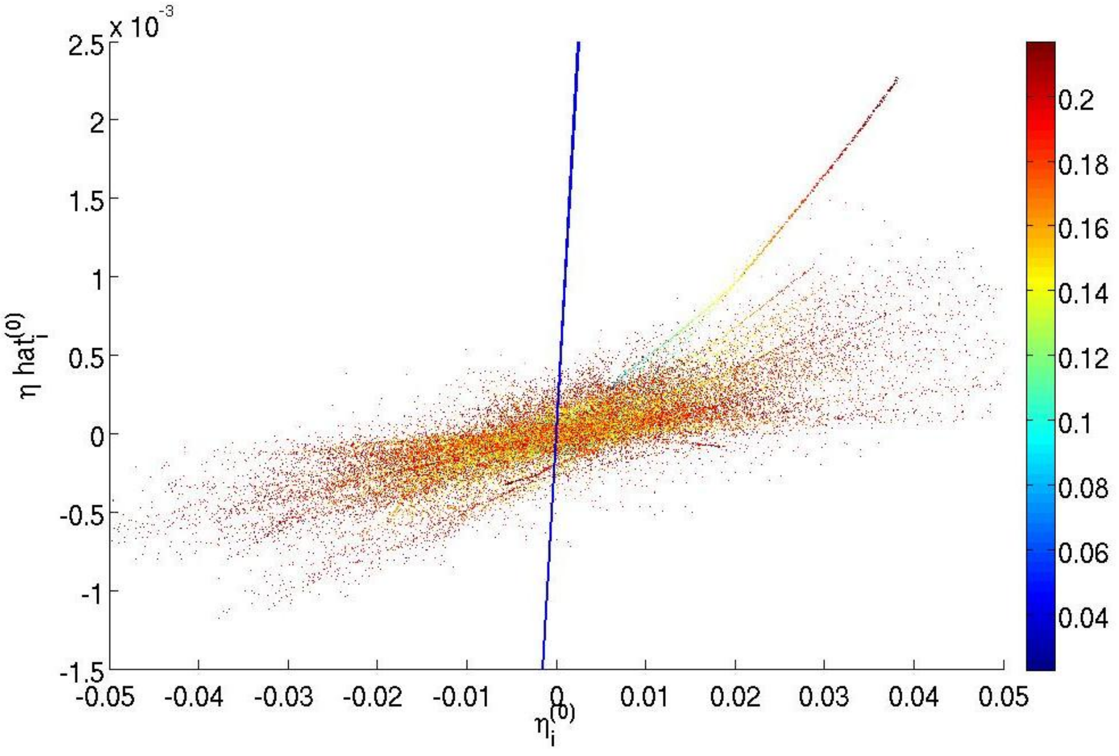
Third, I use multiple realizations of $\Delta\boldsymbol{m}^{(\alpha)}$ and $\boldsymbol{\eta}^{(\alpha)} = \mathbf{A}\cdot\Delta\boldsymbol{m}^{(\alpha)}$ to calculate $\langle\boldsymbol{\eta}^{(\alpha)}|\boldsymbol{\delta}\rangle$ and $\langle\Delta\boldsymbol{m}^{(\alpha)}|\boldsymbol{\delta}\rangle$ for each $\delta^{(\alpha)} = \kappa^{(\alpha)} + \eta^{(\alpha)} + \zeta^{(\alpha)}$. I assume a default guess of 0 for the systematic error in each cell and define the *default error* as $\left|\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}\right|$. I compare this against the *estimate error* $\left|\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})} - \langle\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}|\boldsymbol{\delta}\rangle\right|$. For each cell, I average each set of errors and see whether the estimate is closer to the true shot noise than zero.

Finally, I do the same thing for the photometric zero-points in each SEGMENT. I assume a default guess of 0 for the zero-point in each SEGMENT and define the *default error* as $\left|\Delta m_i^{(\boldsymbol{\alpha})}\right|$. I compare this against the *estimate error* $\left|\Delta m_i^{(\boldsymbol{\alpha})} - \langle\Delta m_i^{(\boldsymbol{\alpha})}|\boldsymbol{\delta}\rangle\right|$. For each SEGMENT, I average each set of errors and see whether the estimate is closer to the true zero-point than zero.

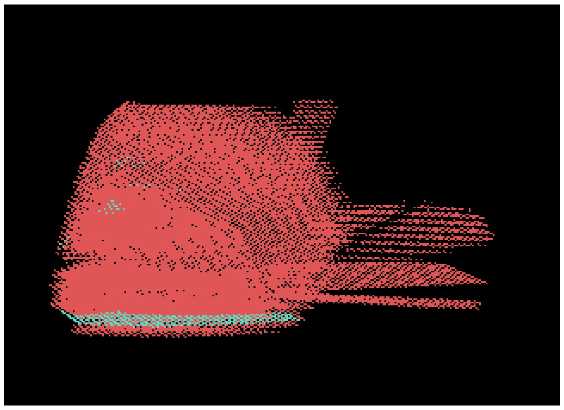
$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \|\boldsymbol{d}\|, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n\rangle, \delta$	base/Overdensities_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR
$RegionID, L_{seg}$	base/PrimarySegmentLengths_R[r].txt
$\mathbf{Z}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\kappa)}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\mathbf{A}$	cells_grids/A_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR

$\mathbf{\Sigma}_{\kappa\eta}^{-1}\mathbf{B}$	g…/clean/Sigma_KappaEta_inv_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\mathbf{B}$	clean/Eigen_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$\boldsymbol{\lambda}^{(B)}$	clean/Eigen_B_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
clean/egd.m ~	
$\eta_i^{(0)}, \hat{\eta}_i^{(0)}, z$	clean/EtaTestSingle_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_[nr].txt
$\Delta\boldsymbol{m}^{(0)}, \Delta\widehat{\boldsymbol{m}}^{(0)}, \Delta\widetilde{\boldsymbol{m}}^{(0)}, l_{SEG}$	clean/DmTestSingle_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_[nr].txt
$\langle\left\|\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}\right\|\rangle, \langle\left\|\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})} - \langle\eta_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}\|\boldsymbol{\delta}\rangle\right\|\rangle, z$	clean/EtaTestMultiple_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_[nr].txt
$\langle\left\|\Delta m_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}\right\|\rangle, \langle\left\|\Delta m_{\boldsymbol{i}}^{(\boldsymbol{\alpha})} - \langle\Delta m_{\boldsymbol{i}}^{(\boldsymbol{\alpha})}\|\boldsymbol{\delta}\rangle\right\|\rangle, l_{SEG}$	clean/DmTestMultiple_[sp02]_R[r]_[z_{min}]_[z_{max}]_bp62_[nr].txt

The general outcome of this investigation is that this algorithm can pick out the effects of the systematic noise, but at the magnitude of $\sigma_m = 0.02$, the results are weak. Consider these results for the R8 cells. The blue line indicates the linear trend, but the correlation between the truth and the estimate is only weakly positive.

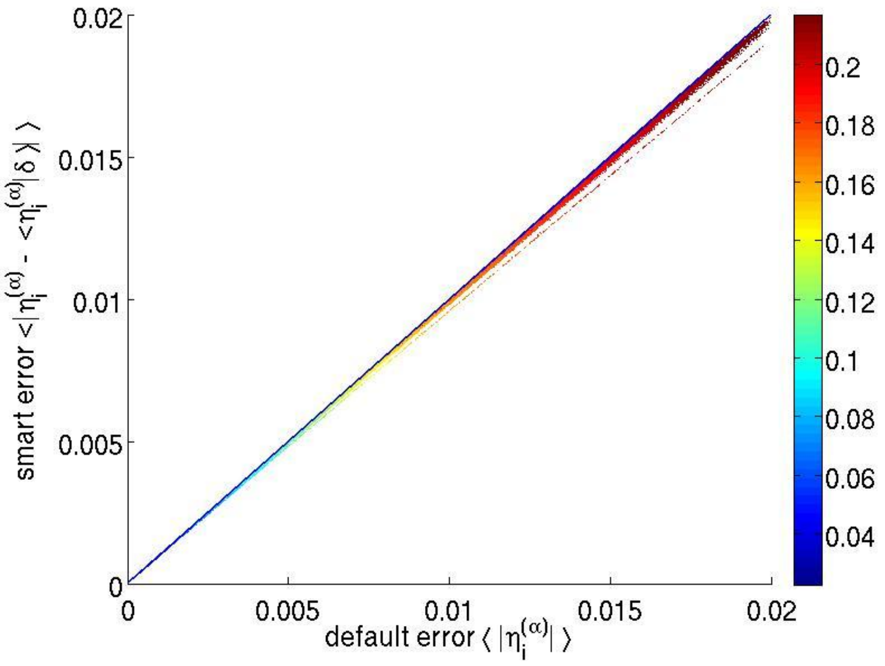


However, you still get the sense here that this is doing a better job than nothing at all. The predicted systematic error overdensities are still trending the right way and there are isolated “strings” of cells that appear to be more positively sloped than others. I conducted a visual investigation of these cells along the most prominent string in the upper right-hand corner of the figure. I find that they are aligned along the longest SEGMENT in the survey. (This was a quick and dirty cut so the other blue dots off of the main SEGMENT are almost certainly not related.)

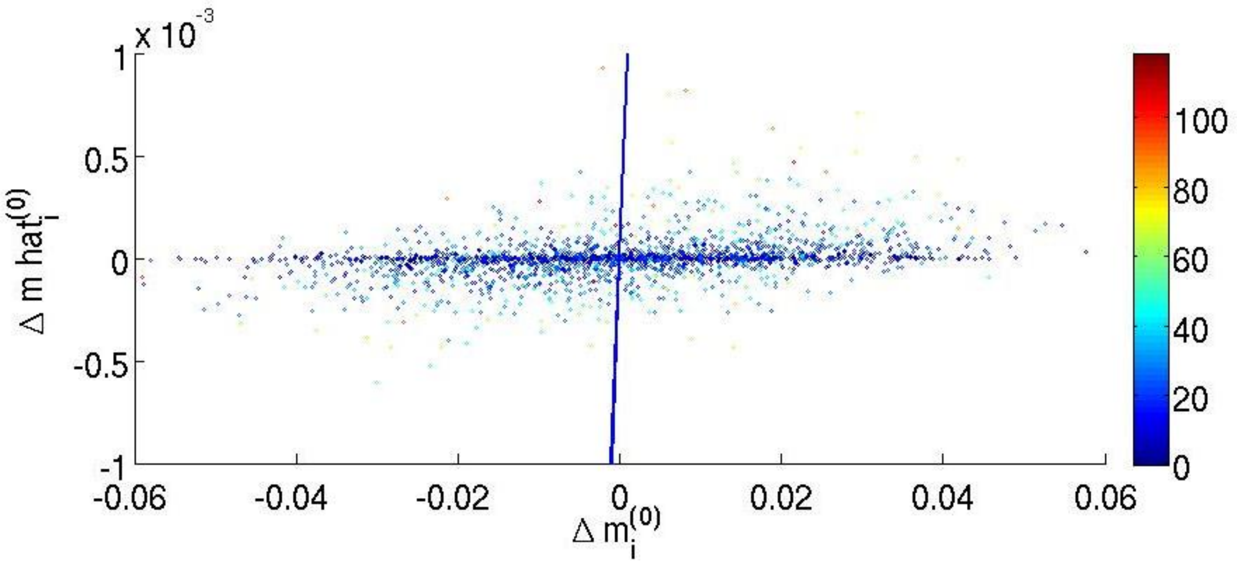


This indicates that prediction of the systematic error is possible, but difficult especially in the presence of other higher magnitude sources of signal and noise. However, the predictive capacity of the estimator is better for groups of cells that are subjected to the same systematic offset.

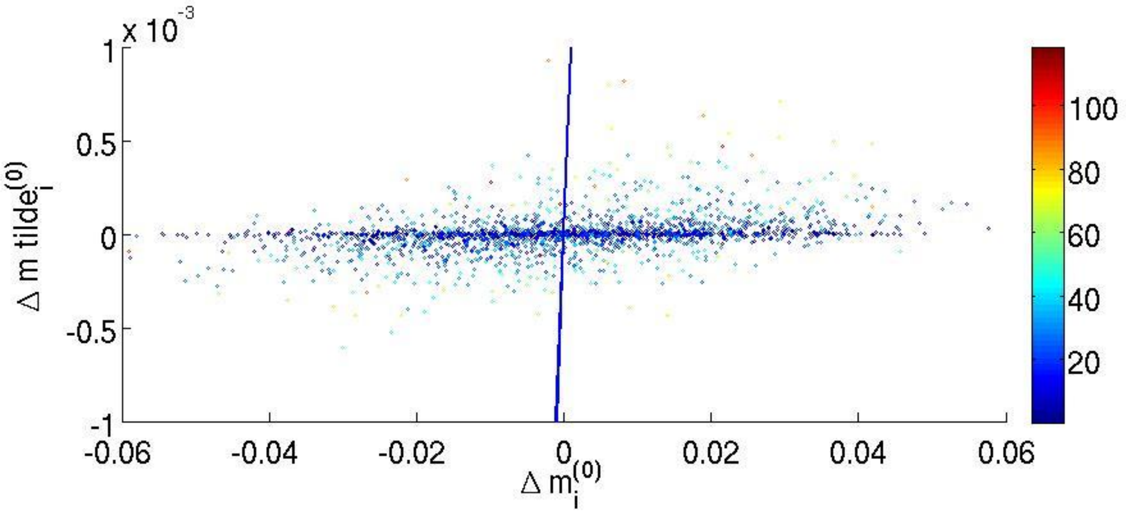
When we allow the systematic overdensities to vary we find that our estimator on average does offer a better prediction than would a default guess of $\Delta \boldsymbol{m} = 0$. As with the shot noise case, the estimate is more powerful for high redshift cells. One also notices a separated string below the main line of points. Again, this refers to cells in the longest SEGMENT of the footprint.



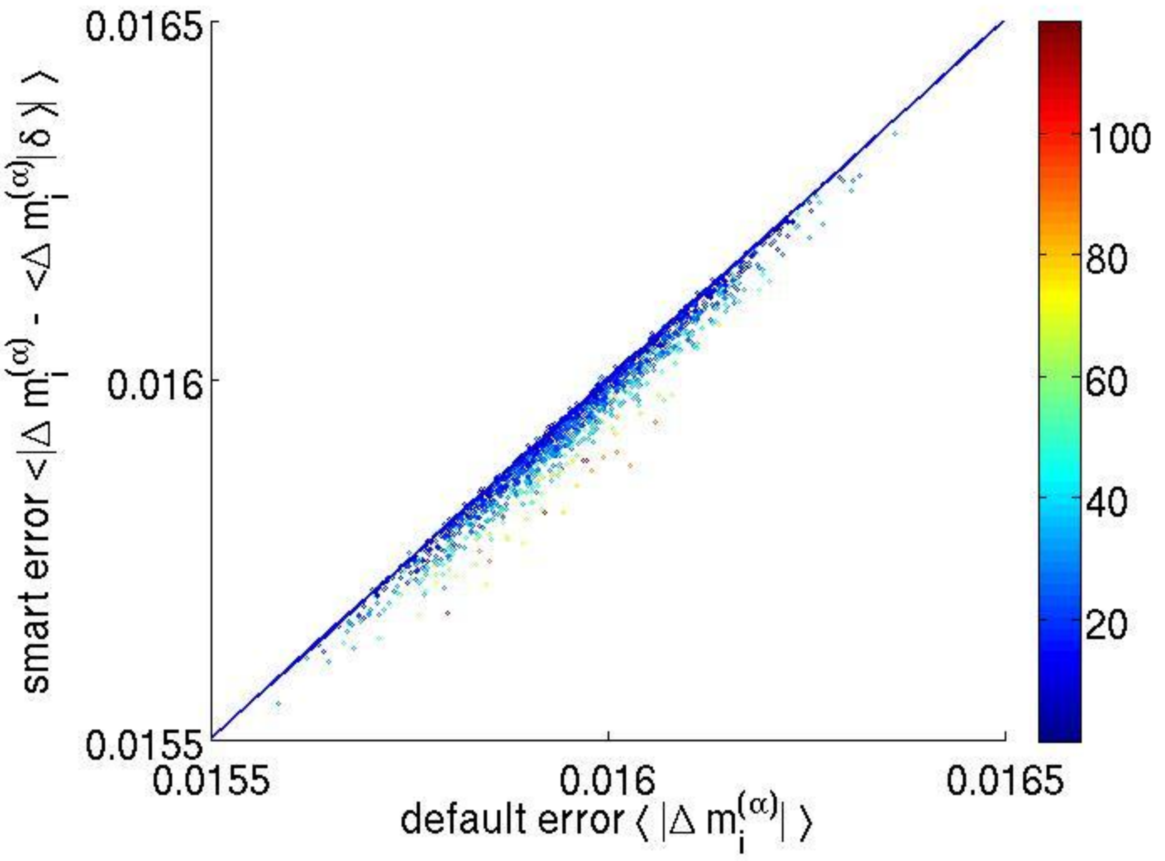
In looking at the solution for $\Delta\widehat{\boldsymbol{m}}^{(0)}$, I find that as with $\widehat{\boldsymbol{\eta}}^{(0)}$ the predictive power of the zero-points is weak, but present. The colorbar is the length of the associated PRIMARY SEGEMNT in degrees. Most SEGMENTs are too small, and therefore contain too little information, for our algorithm to say anything useful about their zero-points. This is shown through the nearly horizontal line of blue dots for which $\Delta\widehat{m}_i^{(0)} \approx 0$. As the SEGMENTs get longer (visually this is the light blue into red) then start to gradually approach the linear trend that represents a perfect prediction. Only SEGMENTs with non-zero lengths are plotted here.



I get the same answer from both $\Delta\widehat{\boldsymbol{m}}^{(0)}$ and $\Delta\widetilde{\boldsymbol{m}}^{(0)}$. That's encouraging and lowers the amount of work I'd have to do in figuring out which is preferable.



When the signal, shot noise and zero-point noise realizations are permitted to vary, the predicted zero-points have the behavior as visualized in the figure below.



We again see that on average my estimator performs better than the default position of simply taking all of the zero-points to be equal to zero. As before, the better performance is experienced by the longer SEGMENTs.

Analysis of Power Spectrum

We want to examine the effect cleansing has on the galaxy clustering power spectrum. We will do this in two ways – by 1) plotting an actual power spectrum by averaging power over discretized k bins and 2) integrating over all length scales to determine the total variance.

Power in Bins

Let's start by figuring out the power spectrum in discretized bins. There are a couple different ways to do this, but I will employ a non-parametric estimator that utilizes the signal eigenmodes. I established earlier that the power spectrum of the clustering signal can be represented as a variance weighted sum over all N Fourier-transformed signal eigenvectors $\hat{\boldsymbol{z}}$,

$$P_\kappa(\boldsymbol{k}) = \langle|\delta_\kappa(\boldsymbol{k})|^2\rangle = \sum_{m=1}^{N} \lambda_m^{(\kappa)} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2.$$

Because the eigenvectors $\hat{\boldsymbol{z}}$ are discretized, the best we can do in finding each $\hat{\boldsymbol{z}}_m(\boldsymbol{k})$ is a fast Fourier transform. This means we will have to split k-space into a discrete number of (about 480 or so) bins. Let the boundaries of the n^{th} bin be k_n and k_{n+1}. The power in this bin is the average of the Fourier amplitudes between those two band powers,

$$P_{\kappa,n} = \sum_{m=1}^{N} \lambda_m^{(\kappa)} \langle|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2\rangle_n,$$

where

$$\langle|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2\rangle_n = \begin{cases} \langle|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2\rangle & k_n \le |\boldsymbol{k}| < k_{n+1} \\ 0 & \text{otherwise} \end{cases}.$$

Using the eigenvalues $\lambda_m^{(\kappa)}$ will recover the fiducial power spectrum used in our model. But to assess the quality of the signal reconstruction, we should replace this with the variance of the mean-zero estimated signal coefficients in signal-space such that

$$P_{\kappa,n} = \sum_{m=1}^{N} \hat{s}_m^2 B_{mn},$$

where $B_{mn} \equiv \langle|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2\rangle_n$ and $\hat{\boldsymbol{s}} = \mathbf{Z}^T\hat{\boldsymbol{\kappa}}$. Note that once the signal modes and Fourier grid spacings are set, the B coefficients need only be solved for once. New estimated signal coefficients can be plugged in thereafter to yield their unique power spectrum.

To find the variance of the power in each bin I begin with the knowledge that each of the B_{mn} terms are independent of one another. This holds since the eigenmodes from which they derive are linearly independent. Therefore,

$$\mathrm{Var}(P_{\kappa,n}) = \mathrm{Var}\left(\sum_{m=1}^{N} \hat{s}_m^2 B_{mn}\right)$$

$$= \sum_{m=1}^{N} \mathrm{Var}(\hat{s}_m^2 B_{mn})$$

$$= \sum_{m=1}^{N} \hat{s}_m^4 \mathrm{Var}(B_{mn}).$$

B_{mn} is an average, so the variance on the average will be the variance of all the terms that determine the coefficient (this is what is given in `d…/power_spectra/coef/Bmn[`n`]_R[`r`]_kappa_Var1B_fwriteR`). I should take the values in that table and divide each by $\sqrt{n_n}$ where n_n is the number of Fourier modes that contributed to the average. This value is reported in the final column of the table.

At times it is more convenient to represent the power in terms of the W-space eigenmodes.

$$P_\kappa(\boldsymbol{k}) = \langle |\delta_\kappa(\boldsymbol{k})|^2 \rangle = \sum_{m=1}^{N} \langle \mathcal{s}_m^2 \rangle |\widehat{\boldsymbol{w}}_m(\boldsymbol{k})|^2.$$

The best estimate of the power in the nth bin is

$$P_{\kappa,n} = \sum_{m=1}^{N} \langle \mathcal{s}_m^2 | \delta \rangle \, D_{mn},$$

when $D_{mn} \equiv \langle |\widehat{\boldsymbol{w}}_m(\boldsymbol{k})|^2 \rangle_n$.

Integrated Power

We can measure the total signal variance σ^2 by integrating over the entire power spectrum,

$$\sigma^2 = \frac{1}{(2\pi)^3} \int_0^\infty d^3k \;\; P_\kappa(\boldsymbol{k}).$$

Within a finite Fourier box, the best we can do is a sum over the available $\boldsymbol{k}$-vectors,

$$\sigma^2 = \frac{1}{(2\pi)^3} \sum_{\boldsymbol{k}} (\Delta k)^3 \; P_\kappa(\boldsymbol{k}),$$

where Δk is the resolution of the grid boxes in Fourier space. This is equivalent to the following when V is the volume of the full Fourier grid in cell-space,

$$\sigma^2 = \frac{1}{V} \sum_{\boldsymbol{k}} P_\kappa(\boldsymbol{k}).$$

For my problem, this is almost always set to $V = (2480\; h^{-1}Mpc)^3$. Note that this is a finite sum over all 512^3 $\boldsymbol{k}$-vectors. It follows that

$$\sigma^2 = \frac{1}{V} \sum_{\boldsymbol{k}} \sum_{m=1}^{N} \lambda_m^{(\kappa)} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2$$

$$= \frac{1}{V} \left[\sum_{m=1}^{N} \lambda_m^{(\kappa)} \sum_{\boldsymbol{k}} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2 \right].$$

Again, since we are assessing the impact of the estimated signal, we replace the eigenvalue with the square of the signal coefficient. Therefore the total signal variance can be written like so when $C_m \equiv \sum_{\boldsymbol{k}} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2$. These can be precomputed and stored once the signal modes are finalized.

$$\sigma^2 = \frac{1}{V} \sum_{m=1}^{N} \hat{s}_m^2 \; C_m.$$

To perform the FFTs I use the FFTW algorithm implemented through MATLAB. Note that this algorithm does *not* normalize the transforms. In other words, transforming a function from Fourier space then back again will yield a result that differs from the original function by a factor of $(2\pi)^{-3}$. As such, the numerical values of $|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2$ will ultimately be scaled by $(2\pi)^{-3/2}$ to be correct.

Solving for the B_{mn} and C_m Coefficients

Fourier transforming one signal mode requires approximately 3 minutes on a single processor. Because there are tens of thousands of modes, the full suite of transforms is quite expensive. As a result, it pays to only have to do it once. In this section I describe the code that gets this job done.

Unlike the C_m coefficients ,which only depend on the signal eigenvectors and the resolution of the Fourier grid, the B_{mn} coefficients also depend upon how many bins you choose to divide your power spectrum plot into. It's difficult to know exactly what number will be best a priori, so the code I outline below is flexible enough to solve for a suite of k-bins. The timings below assume breaking into 3 different sets of k-bins (6 total if you count both log-bins and equal-number-in-each bins).

$\mathbf{Z}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_Vectors_fwriteC
$k_x, k_y, k_z, \|\boldsymbol{k}\|, sd$	cells_grids/kg_[n_x]_[L]_fwriteR
$Cell\ ID, gridbox, fraction$	cells_grids/cgf_R[r]_[z_{min}]_[z_{max}]_[β_{SPEC}]_[512]_[1240]_fwriteR
ind	base/CMATLAB_fftmap_[512]_fwriteC
$Cellcg$	base/Cellcg_R[r]_[512]_[1240]_fwriteR
power_spectra/FFTEigenmodes.m ~ 3.12 min per R8 mode per processor ~ 2.5 days for R8 on 40 processors ~ 18,500 R8 modes per day on 40 processors ~ 4.4 min per R7 mode per processor ~ 5.3 days for R7 on 40 processors ~ 18,000 R7 modes per day on 40 processors	
B_{mn}	power_spectra/coef/Bmn[n]_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/coef/Bmn[n]_R[r]_kappa_[n_{bins}]_fwriteR
C_m	power_spectra/coef/Cm[n]_R[r]_kappa_[512]_[1240]_fwriteC

The first output B_{mn} file contains coefficients where the bins are equally spaced in log-space. The second output file contains coefficients where the bins each have an approximately equal number of $\boldsymbol{k}$-vectors in each bin.

A limitation in the MATLAB parallelization toolbox restricts the number of processors per window to 12. To run on 40 processors, I split my operations into 4 groups of 10. I run the code for a day, save the output then run for another day and a half to protect against lost work. I place a value of $n = 1,2,3$ or 4 in front of the Bmn to indicate which of the 4 files I'm dealing with. Ultimately these files are put back together again using the script below.

B_{mn}	power_spectra/coef/Bmn[n]_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/coef/Bmn[n]_R[r]_kappa_[n_{bins}]_fwriteR
C_m	power_spectra/coef/Cm[n]_R[r]_kappa_[512]_[1240]_fwriteC
power_spectra/AggregatePower.m ~ couple of seconds	
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_kappa_[n_{bins}]_fwriteR
C_m	power_spectra/Cm_R[r]_kappa_[512]_[1240]_fwriteC

A stripped-down, more efficient version of this algorithm is used for R6. It is aggregated in the usual way using the routine referenced above.

$\mathbf{Z}$	`signal/Eigen_kappa_R6_0p02_0p22_bp62_Vectors_fwriteC`
$k_x, k_y, k_z, \|\boldsymbol{k}\|, sd$	`cells_grids/kg_512_1240_fwriteR`
$Cell\ ID, gridbox, fraction$	`cells_grids/cgf_R6_0p02_0p22_bp62_512_1240_fwriteR`
ind	`base/CMATLAB_fftmap_512_fwriteC`
$Cellcg$	`base/Cellcg_R6_512_1240_fwriteR`
power_spectra/FFTEigenmodesR6.m ~ 3.9 min per mode per processor ~ 7.5 days on 40 processors ~ 14,600 modes per day on 40 processors	
B_{mn}	`power_spectra/coef/Bmn[`n`]_R6_kappa_l480_fwriteR`
C_m	`power_spectra/coef/Cm[`n`]_R6_kappa_512_1240_fwriteC`

I wrote R6 specifically to create 480 bins with the idea that I could clump adjacent bins together to get other divisions like 30 bins or 48 bins. More on this is written about below.

A slimmed down version of this algorithm can be used to evaluate the spectral weighting coefficients for the W-space eigenmodes. Here the number of bins is fixed at 30 and only the D_{mn} coefficients for the logarithmically spaced k values are returned.

$\mathbf{W}$	`clean/Eigen_W_[sp02]_R[`r`]_[`z_{min}`]_[`z_{max}`]_bp62_Vectors_fwriteC`
$k_x, k_y, k_z, \|\boldsymbol{k}\|, sd$	`cells_grids/kg_[`n_x`]_[`L`]_fwriteR`
$Cell\ ID, gridbox, fraction$	`cells_grids/cgf_R[`r`]_[`z_{min}`]_[`z_{max}`]_[`β_{SPEC}`]_[512]_[1240]_fwriteR`
ind	`base/CMATLAB_fftmap_[512]_fwriteC`
$Cellcg$	`base/Cellcg_R[`r`]_[512]_[1240]_fwriteR`
power_spectra/FFTEigenmodesW.m ~ 6hrs for R16 on 40 processors ~ 22hrs for R8 on 40 processors ~ ? days for R7 on 40 processors	
D_{mn}	`power_spectra/coef/Dmn[`n`]_R[`r`]_W_l[030]_fwriteR`
E_m	`power_spectra/coef/Em[`n`]_R[`r`]_W_[512]_[1240]_fwriteC`

The `AggregatePower.m` routine is still used for file combination. (See the bottom of the code.)

power_spectra/AggregatePower.m ~ couple of seconds	
D_{mn}	`power_spectra/Dmn_R[`r`]_W_l[030]_fwriteR`
E_m	`power_spectra/Em_R[`r`]_W_[512]_[1240]_fwriteC`

The FFT code in C is constructed to only report the unique $n_x n_y (n_z/2 + 1)$ Fourier vectors while the code in MATLAB returns the full (and largely redundant) set of n_x^3 Fourier vectors. I was unable to fully figure out how the two relate to one another, however. Therefore my approach has been to filter the

MATLAB output until it is identical in structure to the C output and then deal with it as I would have otherwise.

In C, the ordering in each dimension goes like this:

$$k_x[n] = 2\pi\left(\frac{n}{L_x}\right), \quad n = 0,1,2,\ldots,\frac{n_x}{2}-1,\pm\frac{n_x}{2},-\left(\frac{n_x}{2}-1\right),\ldots,-2,-1.$$

The x, y and z indices all start at zero. With the x and y indices held fixed, the z indices vary most quickly. They start at 0 and continue until $\pm n_z/2$. The negative side of the indices are the conjugates of the positive side so they are omitted (though must be doubled) in the C output. The y index is incremented to 1, the z indices are run through, the y index goes to 2 and so on. The x index is incremented to 1 only after the y index of -1 has concluded. The *kg* table referenced in the table above contains this ordering explicitly. The column *sd* equals 1 if the associated $\boldsymbol{k}$-vector is unique and 2 if its complex conjugate exists but is omitted.

In MATLAB this ordering is largely maintained. By means of illustration, here's how the ordering for $n = 8$ grid boxes in each dimension would look like to start.

MATLAB	n_x	n_y	n_z	C
1	0	0	0	1
2	0	0	1	2
3	0	0	2	3
4	0	0	3	4
5	0	0	4	5
6	0	0	-3	
7	0	0	-2	
8	0	0	-1	
9	0	1	0	6
10	0	1	1	7
⋮	⋮	⋮	⋮	⋮

The values in white are those returned in C and referenced in the *kg* table. The spaces in gray are filled in MATLAB but I can't figure out with what. So my first task was to create a mapping between the MATLAB and C coefficients.

Each contiguous "block" of C coefficients has a size of $(n/2+1)$. In all, there are n^2 of these blocks for a total of $n^2(n/2+1)$ MATLAB coefficients I need to keep track of. In this particular case I would want the coefficients $ind =$ [1,2,3,4,5,9,10,11,12,13,17,…]. In other words, if I only call elements ind from the output Fourier table in MATLAB, I should exactly recreate the result I would get in C.

Boundaries of k-bins Equally Spaced in Log-Space

The next step is deciding upon the appropriate boundaries of the k-bins. My first approach is to equally space them in log-space. Storage is a little tricky since there can be x different bin splittings and I didn't want to create x separate tables and code them all by hand.

What I do instead is stack them on top of one another. Here's an example of what the $kbins$ table looks like when my bins are split into 4, 6 and 8 respectively.

$\boldsymbol{kbins}$		
$\boldsymbol{k_{start}}$	$\boldsymbol{k_{end}}$	$\boldsymbol{k}$
0.0025	0.0115	0.0070
0.0115	0.0531	0.0323
0.0531	0.2444	0.1488
0.2444	1.1246	0.6845
0.0025	0.0069	0.0047
0.0069	0.0192	0.0130
0.0192	0.0531	0.0361
0.0531	0.1470	0.1000
0.1470	0.4065	0.2768
0.4065	1.1246	0.7656
0.0025	0.0053	0.0039
0.0053	0.0115	0.0084
0.0115	0.0247	0.0181
⋮	⋮	⋮

Next I need to figure out which $\boldsymbol{k}$-vectors belong in which bins. The kg table has a list of modes that can easily be searched over to find which modes lie within any particular boundary.

The question is how to save this information. I have ordered the $\boldsymbol{k}$ in ascending order and saved them in the column vector y. I save the corresponding C indices of the ordered $\boldsymbol{k}$'s to the column vector ord. Then, using the limits in $kbins$, I go through each row one at a time and find the maximum value of k that is <= k_{end}. I mark the index (of ord) that belongs to this maximum and stick it into the table below. Of course, the first index must always be 1 and the final index must always be $n^2(n/2+1)$.

$\boldsymbol{nbl}$			
1	229	0.0070	389
230	20,005	0.0323	38,641
20,006	1,896,493	0.1488	3,763,737
1,896,494	67,371,008	0.6845	130,414,961
1	51	0.0047	81
52	1008	0.0130	1839
1009	20,005	0.0361	38,641
20,006	414,535	0.1000	818,497

414,536	8,697,581	0.2768	17,314,253
8,697,582	67,371,008	0.7656	116,044,417
1	23	0.0039	33
24	229	0.0084	389
230	2090	0.0181	3887
⋮	⋮	⋮	

The third column of nbl is identical to the third column of $kbins$. For example, if I wanted to reference all of the $\boldsymbol{k}$ information for the elements where $0.1470 < |\boldsymbol{k}| \leq 0.4065$, I would type `kg(ord(414536:8697581),:)`. The fourth column contains the number of $\boldsymbol{k}$-vectors that exist within the specified range of indices (i.e. k-values, but not including k=0). The number includes the vectors doubled through sd. I added this fourth column after first running this routine for R8, so there's a hack in `power_spectra/nbl_fixer.m` that fixes the files.

Boundaries of k-bins with Equal Number of Vectors in Each Bin

The other way to split the $\boldsymbol{k}$-vectors into bins is by placing an equal number of vectors in each bin. As mentioned in the previous section, I have ordered the $\boldsymbol{k}$ in ascending order and saved the corresponding C indices to the long column vector ord. Then I perform a cumulative sum over the corresponding sd elements to learn how many vectors have $\boldsymbol{k}$ magnitudes equal to or less than the value of kg currently referenced. The first and last indices are saved like so. As before, the first index must always be 1 and the final index must always be $n^2(n/2+1)$.

$\boldsymbol{nbn}$		
1	16,840,099	0.3802
16,840,100	33,654,260	0.5778
33,654,261	50,488,942	0.6932
50,488,943	67,371,008	0.8409
1	11,232,801	0.3321
11,232,802	22,445,818	0.5048
22,445,819	33,654,260	0.6001
33,654,261	44,876,120	0.6742
44,876,121	56,104,042	0.7531
56,104,043	67,371,008	0.8738
1	8,428,224	0.3018
8,428,225	16,840,099	0.4586
16,840,100	25,248,224	0.5453
⋮	⋮	⋮

The third column is the average of all the $\boldsymbol{k}$ magnitudes within the specified bin. For example, if I wanted to reference all of the $\boldsymbol{k}$ information for the elements in the 4th of 6 bins, I would type `kg(ord(33654261:44876120),:)`.

Splitting Eigenmode Elements Across Grid Boxes and Fourier Transforming

The next step is to split each eigenvector element fractionally amongst the grid boxes its cell intersects. The fastest way to do this, I think, is to identify all of the unique Cell ID's and figure out the range of elements of cgf that belong to that cell. The example below for R8 says that the grid box ID's and cell/grid box fractions associated with the 4th cell (where CellID = 3) can be found between rows 167 and 220 inclusive of the cgf, gb and $frac$ tables.

$\boldsymbol{un}$
0
1
2
3
4
⋮

$\boldsymbol{Cellcg}$	
1	52
53	106
107	166
167	220
221	277
⋮	⋮

The benefit of this system is that it allows me to relatively quickly populate a column vector where the first 52 elements equal $\hat{z}_{m,1}(\boldsymbol{r})$, elements 53 through 106 equal $\hat{z}_{m,2}(\boldsymbol{r})$ and so on. By multiplying this vector element-wise with the elements in (the equally long) $frac$, I have instantly found the fractional value in each gridbox due to that particular mode. I can then add these values to the grid boxes indicated in gb and I'm ready to Fourier transform.

The following subroutine breaks the signal into grid boxes, Fourier transforms it and exports the coefficients as they would be reported in C.

`gridFFT`

C index	n_x	n_y	n_z	$\lvert\boldsymbol{k}\rvert = kg(:,4)$	sd	zk
1	0	0	0	$\boldsymbol{k}_1$	1	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_1)\rvert^2$
2	0	0	1	$\boldsymbol{k}_2$	2	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_2)\rvert^2$
3	0	0	2	$\boldsymbol{k}_3$	2	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_3)\rvert^2$
4	0	0	3	$\boldsymbol{k}_4$	2	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_4)\rvert^2$

5	0	0	4	$\boldsymbol{k}_5$	1	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_5)\rvert^2$
6	0	1	0	$\boldsymbol{k}_6$	1	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_6)\rvert^2$
7	0	1	1	$\boldsymbol{k}_7$	2	$\lvert\hat{\boldsymbol{z}}(\boldsymbol{k}_7)\rvert^2$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$n^2(n/2+1)$	-1	-1	$\pm n/2$	$\boldsymbol{k}_{n^2(n/2+1)}$	1	$\left\lvert\hat{\boldsymbol{z}}\left(\boldsymbol{k}_{n^2(n/2+1)}\right)\right\rvert^2$

Binning Power and Solving for C_m and B_{mn}

Every value of $\boldsymbol{k}_i$ for which $sd = 2$ is actually standing in for two vectors, with the other lying on the flip side of the z-plane. These powers must be doubled to account for the full Fourier space.

This makes $C_m \equiv \sum_{\boldsymbol{k}} \lvert\hat{\boldsymbol{z}}_m(\boldsymbol{k})\rvert^2$ particularly easy to calculate. I merely need to multiply the sd and zk columns element-wise and sum them up.

The $B_{mn} = \langle\lvert\hat{\boldsymbol{z}}_m(\boldsymbol{k})\rvert^2\rangle_n$ coefficients are saved into tables with the following structure,

$\boldsymbol{Bmn_{eq}}$, $\boldsymbol{Bmn\_log}$					
$B_{1,1}$	$B_{2,1}$	$\cdots$	$B_{N,1}$	k_1	n_{k1}
$B_{1,2}$	$B_{2,2}$	$\cdots$	$B_{N,2}$	k_2	n_{k2}
$B_{1,3}$	$B_{2,3}$	$\cdots$	$B_{N,3}$	k_3	n_{k3}
$B_{1,4}$	$B_{2,4}$	$\cdots$	$B_{N,4}$	k_4	n_{k4}
$B_{1,1}$	$B_{2,1}$	$\cdots$	$B_{N,1}$	k_1	n_{k1}
$B_{1,2}$	$B_{2,2}$	$\cdots$	$B_{N,2}$	k_2	n_{k2}
$B_{1,3}$	$B_{2,3}$	$\cdots$	$B_{N,3}$	k_3	n_{k3}
$B_{1,4}$	$B_{2,4}$	$\cdots$	$B_{N,4}$	k_4	n_{k4}
$B_{1,5}$	$B_{2,5}$	$\cdots$	$B_{N,5}$	k_5	n_{k5}
$B_{1,6}$	$B_{2,6}$	$\cdots$	$B_{N,6}$	k_6	n_{k6}
$B_{1,1}$	$B_{2,1}$	$\cdots$	$B_{N,1}$	k_1	n_{k1}
$B_{1,2}$	$B_{2,2}$	$\cdots$	$B_{N,2}$	k_2	n_{k2}
⋮	⋮	⋮	⋮	⋮	

Each column is reserved for the power due to one signal mode. Column $N + 1$ contains the k value for that bin. These values should be the points at which the power spectrum is plotted. The logarithmically spaced bins have an additional column at the end containing the number of k-vectors that went into building that B_{mn} coefficient.

Because the B_{mn} values are averages, they are evaluated somewhat differently than C_m. Using the nb and nbn tables, the $\boldsymbol{k}$—vectors corresponding to each k_i range is referenced. The power in those grid boxes are multiplied element-wise by sd and summed just like in C_m. To get the average, though, one must also divide by the sum of that range's sd values.

The Bmn_log and Bmn_eq tables are subsequently separated and each n_{bins} block is saved to its own file.

Clumping the B_{mn} Coefficients

This will mostly reference the work I did for R6. During the Fourier transforms R6 is split into 480 bins. The output B_{mn} file looks something like this,

	Mode 1	**Mode 2**		**Mode N**	$\boldsymbol{k}$	$\boldsymbol{n_k}$
Bin 1	$B_{1,1}$	$B_{2,1}$	$\cdots$	$B_{N,1}$	k_1	n_{k1}
Bin 2	$B_{1,2}$	$B_{2,2}$	$\cdots$	$B_{N,2}$	k_2	n_{k2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Bin 10	$B_{1,10}$	$B_{2,10}$	$\cdots$	$B_{N,10}$	k_{10}	n_{k10}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Bin 480	$B_{1,480}$	$B_{2,480}$	$\cdots$	$B_{N,480}$	k_{480}	n_{k480}

Let k_0 be the lower limit of the first bin. Let k_1 be the upper limit of the first bin and the lower limit of the 2nd bin. Let k_2 be the upper limit of the 2nd bin and the lower limit of the 3rd bin and so on. Let the averages within the ith bin be denoted b_i. (For the purposes of this exercise ignore the notation given by the table above.) The first bin's average is defined to be

$$b_1 = \frac{k_0 + k_1}{2}.$$

Suppose we are averaging over the first 4 bins. We are looking for the quantity

$$k = \frac{k_0 + k_4}{2}.$$

We can set up a matrix equation

$$\begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \end{bmatrix} = \begin{bmatrix} b_1 - (1/2)k_0 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

This allows us to solve for k_4 and then solve for k. When averaging over the next bin, the last k_4 becomes the new k_0.

Recall that B_{ij} is an average. The total sum that was averaged was $B_{ij}n_{k_j}$ where n_{k_j} is the number of k-vectors that went into the average of bin j. If I was clumping the first 10 bins, I would add all these up and take their average,

$$\frac{\sum_{j=1}^{10} B_{ij}n_{k_j}}{\sum_{j=1}^{10} n_{k_j}}.$$

This is how the new B_{mn} coefficient is defined. With 480 to start with, there would be 48 of these per mode.

B_{mn}	`power_spectra/coef/Bmn_R[6]_kappa_l[480]_fwriteR`
power_spectra/BmnSplit.m ~ 4 seconds each	
B_{mn}	`power_spectra/coef/Bmn_R[6]_kappa_l[030]_fwriteR`

I can also do splitting dynamically. Instead of grouping 48 at a time, I can do consecutive groups with a different number of averaged k modes each time. This can be useful since there is more spectral information available at large k. To average it in the same way as the low k modes would discard information unnecessarily.

The task then is to develop a method of dynamically splitting B_{mn} according to my plotting needs. I'm going to take a cue from Percival (2007) in setting up my splitting. There are about 50 data points $k > 10^{-0.9}$, 8 in $10^{-1} \leq k \leq 10^{-0.9}$, and 10 between $10^{-1.5} \leq k \leq 10^{-1}$. There are 7 more at k values less than that. This leads to a total of 75 points. This breakdown should give a sense of how many k blocks to group together when defining in point on the power spectrum. I think this sort of spacing might work out. The sum of the elements in this vector is 480.

B_{mn}	d…/power_spectra/coef/Bmn_R[r]_kappa_l[480]_fwriteR
power_spectra/BmnSplit2.m ~ few seconds	
B_{mn}	d…/power_spectra/coef/Bmn_R[r]_kappa_l[B]_fwriteR

I have to organize my splitting by letters since they are difficult to characterize by numbers alone. Here's how they go.

A: `sp = [140,40,6,6,4*ones(1,25),2*ones(1,94)];`
B: `sp = [80,60,40,20,10,4*ones(1,24),2*ones(1,87)];`

Plotted Power Spectra

In this section I will introduce two methods by which to plot power spectra using the B_{mn} coefficients – empirically and analytically. I will display the spectra for $\boldsymbol{\delta}$, $\boldsymbol{\delta}_\kappa$, $\boldsymbol{\delta}_\eta$ and $\boldsymbol{\delta}_\zeta$ using both methods and the spectrum for $\hat{\boldsymbol{\kappa}}$ using only the empirical method.

By default, the power spectra coefficient B_{mn} are calculated in signal-space. The code that follows will assume that this default is maintained. (It is possible to do the same thing with the coefficients calculated in another space, like noise-space, but the results *should* be the same and in the interest of time I will forgo any additional investigation or investment in making the code more modular.) Input vectors must be in cell-space. The necessary rotation $\boldsymbol{d} = \mathbf{Z}^T\boldsymbol{\delta}$ into signal-space will be done by the code.

The empirical method involves inputting a collection of actual vectors, finding the power spectrum of each and returning both the individual spectra and their average with error bars. This code assumes the B_{mn} coefficients are of the Bmn_log variety.

Analytical Method

All Power Spectra

Rather than find the power spectra of individual vectors, the analytical method finds the power spectra using the variances of various signal and noise components. To plot their spectra these variances must be computed in signal-space, so let's take each component individually. Recall that a full vector in cell-space $\boldsymbol{\delta} = \boldsymbol{\delta}_\kappa + \boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta$ corresponds to $\boldsymbol{d} = \boldsymbol{d}_\kappa + \boldsymbol{d}_\eta + \boldsymbol{d}_\zeta$ in signal-space.

<u>SIGNAL</u>

The variance of the signal along each dimension of signal-space is trivial,

$$\mathrm{Var}(d_\kappa[m]) = \lambda_m^{(\kappa)}.$$

<u>SHOT NOISE</u>

In signal-space, a shot noise overdensity vector is $\boldsymbol{d}_\zeta = \mathbf{Z}^T\boldsymbol{\delta}_\zeta$. The mth element of $\boldsymbol{d}_\zeta$ equals $d_\zeta[m] = \sum_{i=1}^{N} \mathrm{Z}_{i,m}\delta_\zeta[i]$. The variance of this component is

$$\begin{aligned}\mathrm{Var}\big(d_\zeta[m]\big) &= \sum_{i=1}^{N} \mathrm{Z}_{i,m}^2 \,\mathrm{Var}\big(\delta_\zeta[i]\big)\\ &= \sum_{i=1}^{N} \mathrm{Z}_{i,m}^2 \frac{1}{\langle n_i\rangle},\end{aligned}$$

where $\langle n_i\rangle$ is the expected number of galaxies in the ith cell.

ZERO-POINT NOISE

In signal-space, a zero-point noise overdensity vector is $\boldsymbol{d}_\eta = \mathbf{Z}^T\boldsymbol{\delta}_\eta$ while in noise-space we have $\boldsymbol{\delta}_\eta = \mathbf{U}\,\boldsymbol{\mathfrak{d}}_\eta$. It follows that $\boldsymbol{d}_\eta = \mathbf{M}\boldsymbol{\mathfrak{d}}_\eta$ where $\mathbf{M} = \mathbf{Z}^T\mathbf{U}$. The mth element of $\boldsymbol{d}_\eta$ equals $d_\eta[m] = \sum_{i=1}^{N} \mathrm{M}_{m,i}\mathfrak{d}_\eta[i]$. The variance of this component is

$$\begin{aligned}\mathrm{Var}\big(d_\eta[m]\big) &= \sum_{i=1}^{N} \mathrm{M}_{m,i}^2 \,\mathrm{Var}\big(\mathfrak{d}_\eta[i]\big)\\ &= \sum_{i=1}^{N} \mathrm{M}_{m,i}^2 \lambda_i^{(\eta)}.\end{aligned}$$

DATA VECTOR

Because signal, shot noise and zero-point noise are independent of one another, it follows that

$$\begin{aligned}\mathrm{Var}(\boldsymbol{d}) &= \mathrm{Var}\big(\boldsymbol{d}_\kappa + \boldsymbol{d}_\eta + \boldsymbol{d}_\zeta\big)\\ &= \mathrm{Var}(\boldsymbol{d}_\kappa) + \mathrm{Var}\big(\boldsymbol{d}_\eta\big) + \mathrm{Var}\big(\boldsymbol{d}_\zeta\big)\end{aligned}$$

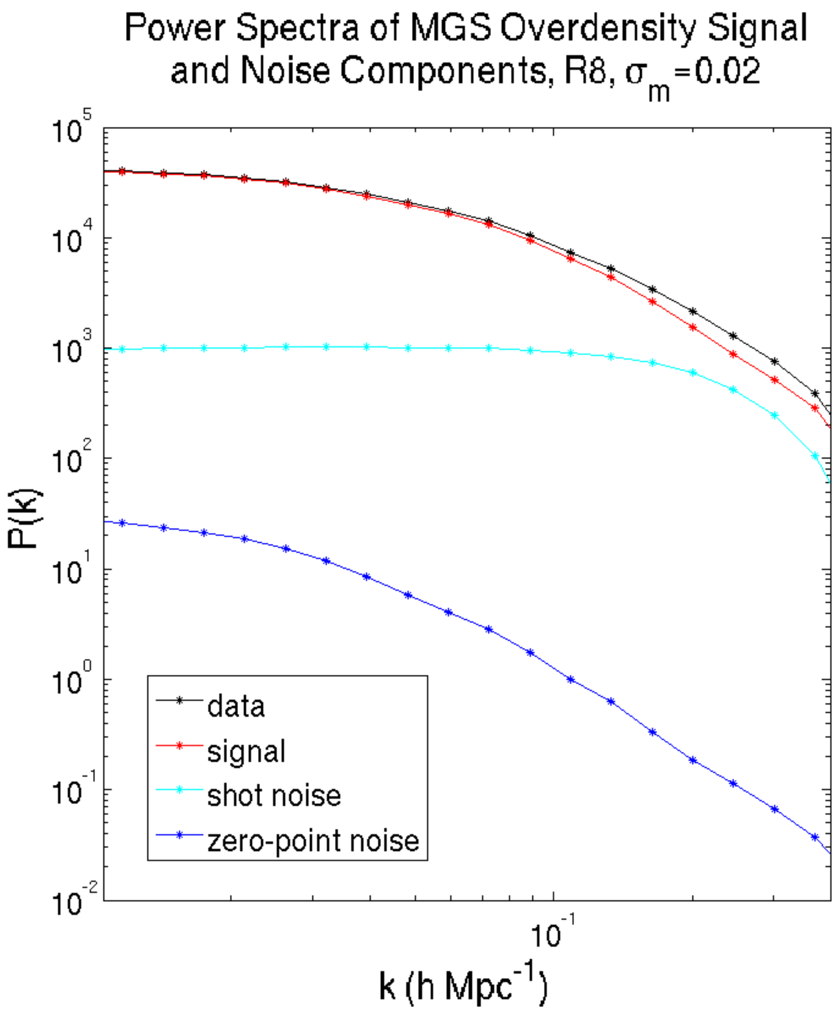
$$\mathrm{Var}(d[m]) = \lambda_m^{(\kappa)} + \sum_{i=1}^{N} \mathrm{M}_{m,i}^2\lambda_i^{(\eta)} + \sum_{i=1}^{N} \mathrm{Z}_{i,m}^2 \frac{1}{\langle n_i\rangle}.$$

Each dimension's variance subsequently scales the B_{mn} coefficients which are then combined to yield power spectra.

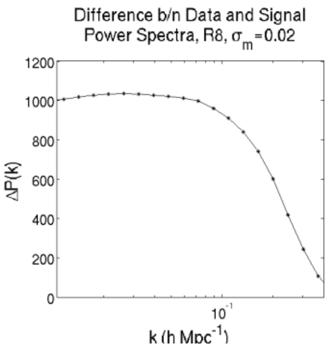
$Cell\ ID, \hat{x}, \hat{y}, \hat{z}, \lvert\boldsymbol{d}\rvert, z, \theta, c, RA_i, RA_f, dec_i, dec_f, \beta_{SPEC}, n, \langle n\rangle, \delta$	base/Overdensities_DR6_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteR
$\boldsymbol{\lambda}^{(\kappa)}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$\boldsymbol{\lambda}^{(\eta)}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Values.txt
$\mathbf{Z}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
$\mathbf{U}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwrite

B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
power_spectra/PkAnalytical.m ~ b/n 15 min and an hour	
$P_\kappa(k), k, n_k$	power_spectra/Pk_kappa_Analytical_R[r]_l[n_{bins}]_fwriteR
$P_\eta(k), k, n_k$	power_spectra/Pk_eta_Analytical_R[r]_[σ_m]_l[n_{bins}]_fwriteR
$P_\zeta(k), k, n_k$	power_spectra/Pk_zeta_Analytical_R[r]_l[n_{bins}]_fwriteR
$P(k), k, n_k$	power_spectra/Pk_data_Analytical_R[r]_[σ_m]_l[n_{bins}]_fwriteR

The following plot displays the power spectra of all three components and their sum.



Here’s a closer look at the difference in power between the raw data and the clustering signal. That the power differs by about 1000 suggests that the principle difference between the raw data and the signal is just shot noise.


Difference b/n Data and Signal
Power Spectra, R8, $\sigma_m = 0.02$
1200
1000
800
600
400
200
0
$\Delta P(k)$
10^{-1}
k (h Mpc^{-1})

This time around I want to incorporate error bars into the power spectrum plots. I reported earlier that

$$P_n = \sum_{m=1}^{N} \hat{s}_m^2 B_{mn},$$

where $B_{mn} \equiv \langle |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2 \rangle_n$ and $\hat{s}_m^2$ is the variance of the component (e.g. signal, zero-point noise, shot noise or data) along that mode. The B_{mn} coefficients are linearly independent. Therefore the variance of the sum equals the sum of the variances,

$$\mathrm{Var}(P_n) = \sum_{m=1}^{N} \mathrm{Var}(\hat{s}_m^2 \, \langle |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2 \rangle_n) = \sum_{m=1}^{N} \hat{s}_m^4 \frac{\mathrm{Var}(|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|_n^2)}{\sqrt{l_n}},$$

where

$$|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|_n^2 = \begin{cases} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2 & k_n \le |\boldsymbol{k}| < k_{n+1} \\ 0 & \text{otherwise} \end{cases},$$

and l_n is the number of $\boldsymbol{k}$-modes for which $k_n \le |\boldsymbol{k}| < k_{n+1}$.

It follows that the variance of the signal power is

$$\mathrm{Var}\left(P_n^{(\kappa)}\right) = \sum_{m=1}^{m} \frac{\lambda_m^{(\kappa)^2} \mathrm{Var}(|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|_n^2)}{\sqrt{l_n}}.$$

The variance of the shot noise power is

$$\mathrm{Var}\left(P_n^{(\zeta)}\right) = \sum_{m=1}^{N} \frac{\left(\sum_{i=1}^{N} \mathrm{Z}_{i,m}^2 \frac{1}{\langle n_i \rangle}\right)^2 \mathrm{Var}(|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|_n^2)}{\sqrt{l_n}}.$$

The variance of the zero-point noise power is

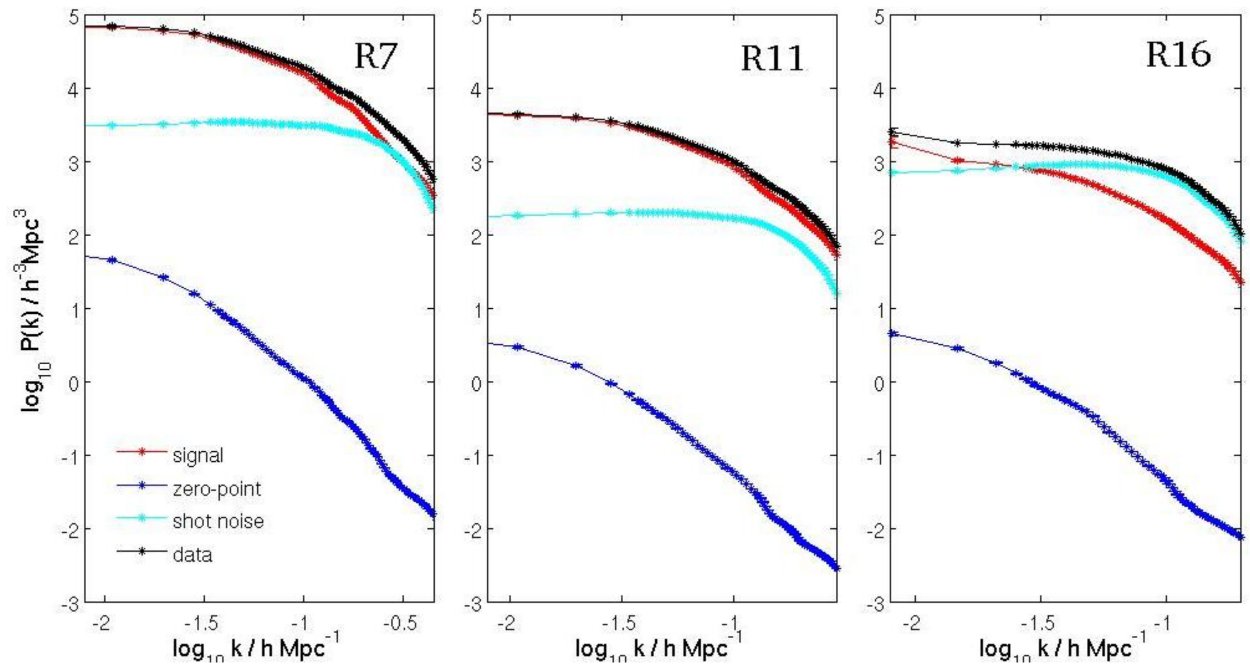
$$\mathrm{Var}\left(P_n^{(\eta)}\right) = \sum_{m=1}^{N} \frac{\left(\sum_{i=1}^{N} \mathrm{M}_{m,i}^2 \lambda_i^{(\eta)}\right)^2 \mathrm{Var}(|\hat{\boldsymbol{z}}_m(\boldsymbol{k})|_n^2)}{\sqrt{l_n}}.$$

The variance of the data is

$$\begin{aligned} \mathrm{Var}\left(P_n^{(\Gamma)}\right) &= \mathrm{Var}\left(P_n^{(\kappa)} + P_n^{(\zeta)} + P_n^{(\eta)}\right) \\ &= \mathrm{Var}\left(P_n^{(\kappa)}\right) + \mathrm{Var}\left(P_n^{(\zeta)}\right) + \mathrm{Var}\left(P_n^{(\eta)}\right). \end{aligned}$$

$\langle n\rangle, n, \delta, n_p, n_i, n_{dr}$	d…/signal/Overdensities_R[r]_fwriteR
$\boldsymbol{\lambda}^{(\kappa)}$	d…/signal/Eigen_kappa_R[r]_Values_fwriteC
$\boldsymbol{\lambda}^{(\eta)}$	d…/noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Values_fwriteR
$\mathbf{Z}$	d…/signal/Eigen_kappa_R[r]_Vectors_fwriteC
$\mathbf{U}$	d…/noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
B_{mn}	d…/power_spectra/coef/Bmn_R[r]_kappa_l[B]_fwriteR
$\mathrm{Var}(\lvert\hat{\boldsymbol{z}}_m(\boldsymbol{k})\rvert_n^2)$	d…/power_spectra/coef/Bmn_R[r]_kappa_Varl[B]_fwriteR
*power_spectra/PkAnalytical2.m ~ b/n 15 min and an hour if **M** is unknown*	
$\mathbf{M}$	d…/power_spectra/M[r]_fwriteC
$k, P_\kappa(k), \sigma_{P_\kappa}, n_{\boldsymbol{k}}$	d…/power_spectra/Pk_kappa_Analytical_R[r]_l[A]_fwriteR
$k, P_\eta(k), \sigma_{P_\eta}, n_{\boldsymbol{k}}$	d…/power_spectra/Pk_eta_Analytical_R[r]_[σ_m]_l[A]_fwriteR
$k, P_\zeta(k), \sigma_{P_\zeta}, n_{\boldsymbol{k}}$	d…/power_spectra/Pk_zeta_Analytical_R[r]_l[A]_fwriteR
$k, P(k), \sigma_P, n_{\boldsymbol{k}}$	d…/power_spectra/Pk_data_Analytical_R[r]_[σ_m]_l[A]_fwriteR

power_spectra/PkPlots.m
plots/thesis/Pk_components



[caption: Analytic power spectra of overdensity components for cells of size 7, 11 and 16 h^{-1} Mpc. Points mark the power averaged over local k-modes. Error bars are one standard deviation of the mean. Powers on scales smaller than the diameter of the cells are excluded from the figures.]

Explaining the Zero-Point Noise Spectrum

We can estimate the shape of the noise power spectrum a priori provided we understand something about the correlation function of the noise. We often approximate the correlation function $\xi(r)$ as the ratio of the number of objects observed to the number expected, at some separation distance r. In a three-dimensional space centered at any point, the expected number of evenly distributed point-pairs at a distance r scales as $\langle N \rangle \sim r^2 dr$.

However, the correlated zero-point noise lies along a stripe, not a spherical shell. When this planar geometry intersects the shell, a ring of radius r results. Zero-point noise clustering is only permitted in this ring where the number of point pairs scales as $r\ dr$. However, the stripe also has a non-zero width which effectively increases the number of pairs to r^p where $p \gtrsim 1$.

The zero-point noise correlation function $\xi_\eta(r)$ may be approximated as

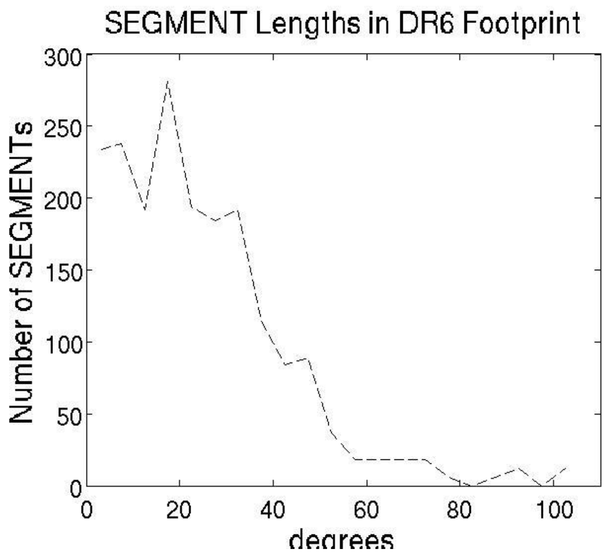
$$\xi_{noise}(r) \propto \frac{N_\eta}{\langle N \rangle} \sim \frac{r^p\ dr}{r^2 dr} \sim r^{p-2}.$$

Translating an isotropic correlation function to a power spectrum occurs in the usual way,

$$P_\eta(k) = 4\pi \int dr\ \ r^2\ \frac{\sin(kr)}{kr} \xi_\eta(r) \sim \frac{1}{k^{p+1}}.$$

From this analysis, we expect $P_\eta(k)$ to go as something between k^{-2} and k^{-3}. As we see in the figure above, this is essentially what results.

We note that the zero-point spectrum flattens at $k \approx 1.5 \times 10^{-2}\ h\ Mpc^{-1}$ or $l \approx 420 h^{-1} Mpc$. This roughly corresponds to the largest scales at which the zero-points still have an impact. The longest SEGMENTs still appreciable in number have lengths of about 50°. At redshifts where $f(z)$ is large, this corresponds to physical distance separations ranging from about $375h^{-1}Mpc$ at $z = 0.15$ to $550h^{-1}Mpc$ at $z = 0.22$. This falls more or less in line with what is revealed by the power spectrum.

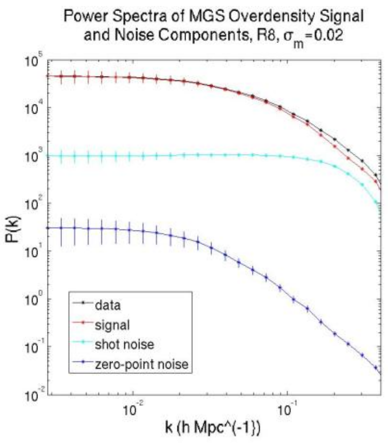


Empirical Method

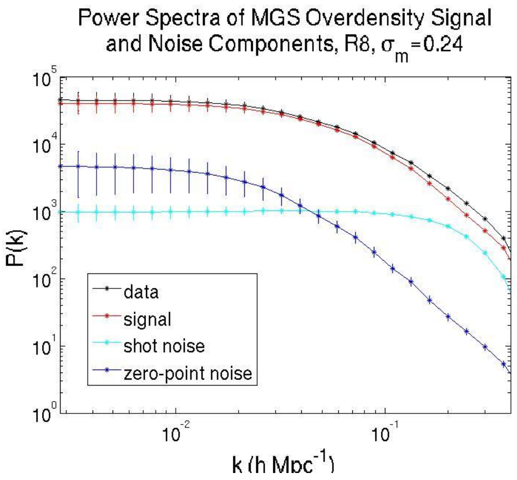
An ensemble of simulated data vectors is used to plot power spectra.

$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\eta$	clean/shot/delta_eta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\widehat{\boldsymbol{\kappa}}$	clean/shot/kappahat_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\mathbf{Z}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
power_spectra/PkEmpirical.m ~	
$P_\kappa(k), k, n_{\boldsymbol{k}}$	power_spectra/Pk_kappa_Empirical_R[r]_l[n_{bins}]_fwriteR
$P_\eta(k), k, n_{\boldsymbol{k}}$	power_spectra/Pk_eta_Empirical_R[r]_[σ_m]_l[n_{bins}]_fwriteR
$P_\zeta(k), k, n_{\boldsymbol{k}}$	power_spectra/Pk_zeta_Empirical_R[r]_l[n_{bins}]_fwriteR
$P(k), k, n_{\boldsymbol{k}}$	power_spectra/Pk_data_Empirical_R[r]_[σ_m]_l[n_{bins}]_fwriteR
$P_{\hat\kappa}(k), k, n_{\boldsymbol{k}}$	power_spectra/Pk_kappahat_R[r]_l[n_{bins}]_fwriteR

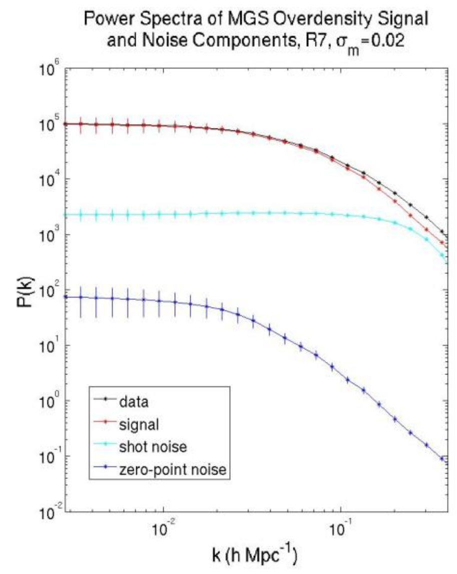
Lk

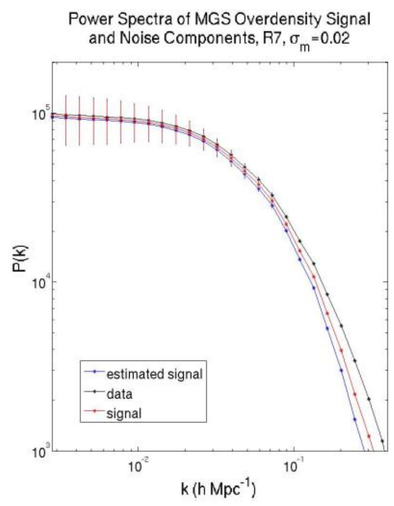


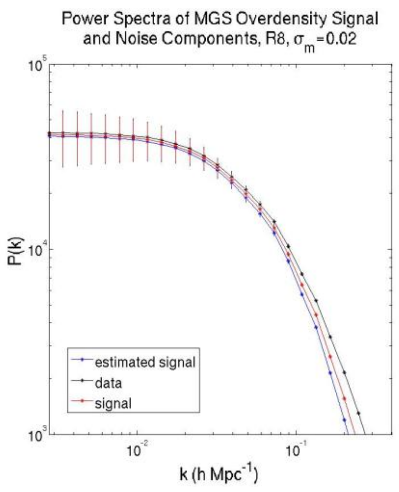
If the magnitude of the zero-point noise increases by a factor of 12, the power of that noise increases by a factor of 144.



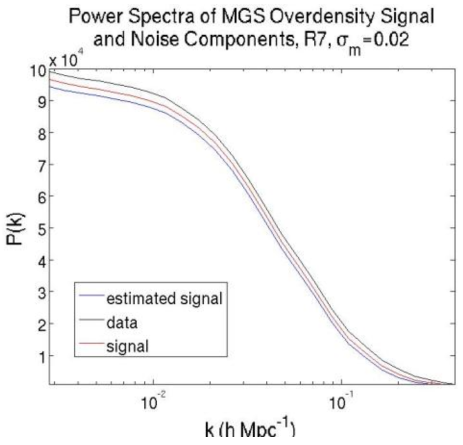
Though the results for R7 are similar, the power does appear to increase in amplitude across the board.

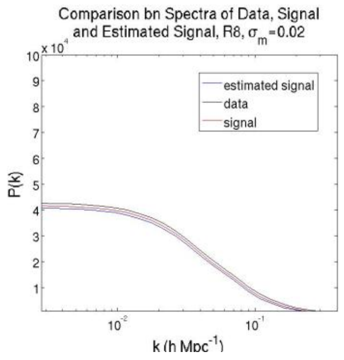




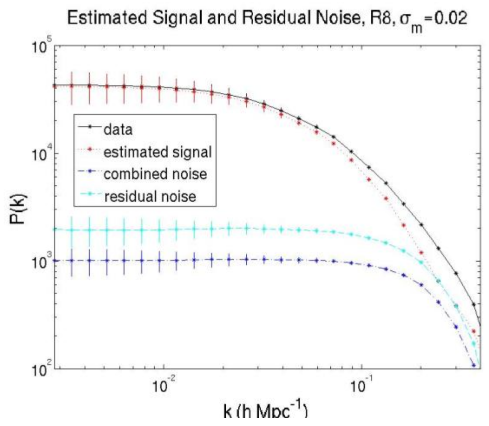


The previous two plots suggest that while the estimated signal is closer to the real signal (in a two-norm sense) than the raw data is to the real signal, the signal estimation method removes a bit too much noise. The fact that the estimated power spectrum lies below the true signal spectrum suggests that some portion of the signal was interpreted to be noise and removed inadvertently. The next two plots illustrate this on a linear scale.





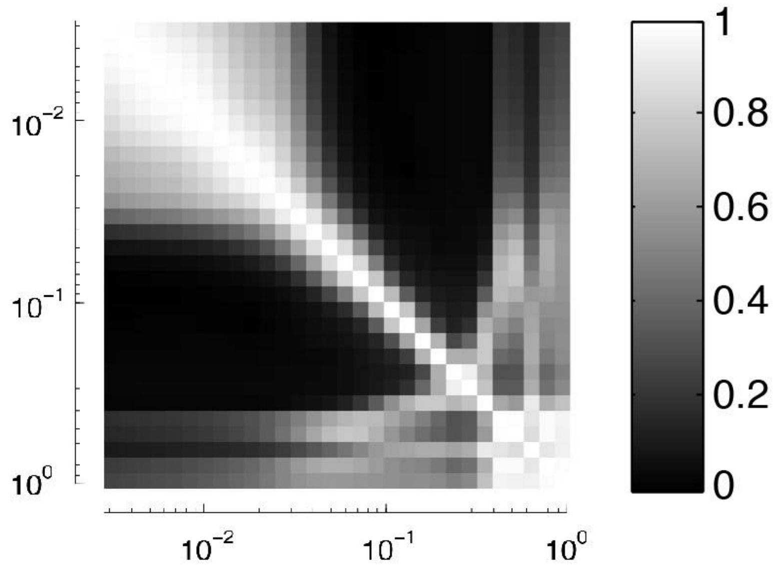
Under ideal circumstances the power of the deprojected noise, $\boldsymbol{\delta} - \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$, would exactly equal the sum of the power of the zero-point and shot noise terms. As shown above, our method removes a bit more noise than is actually present. This leads to $\langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$ somewhat underestimating $\boldsymbol{\kappa}$ and the residual $\boldsymbol{\delta} - \langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle$ somewhat overestimating the actual amount of noise present $\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta$.



We examine the correlations between the band powers of the estimated signal's power spectrum. The figure below shows

$$\mathrm{Corr}_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} C_{jj}}},$$

where $C_{ij} \equiv \mathrm{Cov}\left(P(k_i), P\left(k_j\right)\right)$.



The large scale modes are highly correlated, likely due to the finite survey window. The smallest scale k-modes have sizes on the order of the cell diameters, wherein little information can be gleaned.

Bias Investigation

Power Spectrum Adjustment

My answer in W-space is $\langle s_i|\delta\rangle = x_i/\lambda_i^{(\kappa\nu)}$. Before being saved to file, they are rotated back into cell-space. In terms of evaluating power spectra we rotate into signal-space. To make the notation more concise, let

$$\hat{S} \equiv \langle S|\delta\rangle = \mathbf{Z}^T\langle\kappa|\delta\rangle.$$

While $\langle S|\delta\rangle$ may be the best unbiased solution for the signal in signal-space, $\langle\hat{S}(\mathbf{k})\hat{S}(\mathbf{k})\rangle$ is not the best unbiased solution for its power. To see why, consider that there will always exist a discrepancy ε between the true signal S and its estimate such that $\hat{S} = S + \varepsilon$. The power spectrum of the true signal differs from that of the estimated signal,

$$\begin{aligned}\langle S(\mathbf{k})S(\mathbf{k})\rangle &= \langle\left(\hat{S}(\mathbf{k}) - \varepsilon(\mathbf{k})\right)\left(\hat{S}(\mathbf{k}) - \varepsilon(\mathbf{k})\right)\rangle \\ &= \langle\left|\hat{S}(\mathbf{k})\right|^2\rangle - 2\langle\hat{S}(\mathbf{k})\varepsilon(\mathbf{k})\rangle + \langle|\varepsilon(\mathbf{k})|^2\rangle.\end{aligned}$$

The power in the n^{th} bin is

$$P_{\kappa,n} = \sum_{m=1}^{N} \langle \hat{S}_m^2 \rangle B_{mn} - 2\sum_{m=1}^{N} \langle \hat{S}_m \varepsilon_m \rangle B_{mn} + \sum_{m=1}^{N} \langle \varepsilon_m^2 \rangle B_{mn}.$$

To see where the second term comes from,

$$\hat{S}(\mathbf{k}) = \sum_{m=1}^{N} \hat{S}_m \, \hat{z}_m(\mathbf{k}), \qquad \varepsilon(\mathbf{k}) = \sum_{l=1}^{N} \varepsilon_l \, \hat{z}_l(\mathbf{k}).$$

$$\langle \hat{S}(\mathbf{k})\varepsilon(\mathbf{k}) \rangle = \langle \sum_{m,l}^{N} \hat{S}_m \varepsilon_l \, \hat{z}_m(\mathbf{k}) \hat{z}_l(\mathbf{k}) \rangle.$$

The signal modes are orthonormal and terms survive only when $l = m$. Finally,

$$P_{\kappa,n} = \sum_{m=1}^{N} \left(\langle \hat{S}_m^2 \rangle - 2\langle \hat{S}_m \varepsilon_m \rangle + \langle \varepsilon_m^2 \rangle \right) B_{mn}.$$

This serves as the average power of the unbiased power spectrum $P_{\langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle, unbiased}$ using the signal estimates. I find empirically that $\langle P_\kappa - P_{\langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle, unbiased} \rangle$ equals exactly zero (at least to machine precision). I did so with the code `PkBias.m` described in the next section.

Best Unbiased Power Estimate

The code below creates all of the power spectra needed to show the improvement of the unbiased power derived through $\langle \boldsymbol{s}^2 | \boldsymbol{\delta} \rangle$ over the power derived through $\boldsymbol{\delta}$. A verification that the bias correction introduced in the last section does in fact bring P_κ and $P_{\langle \boldsymbol{\kappa} | \boldsymbol{\delta} \rangle}$ onto equal footing is provided. The error bars given on the differences are the errors *on the mean*, not the errors on the individual differences.

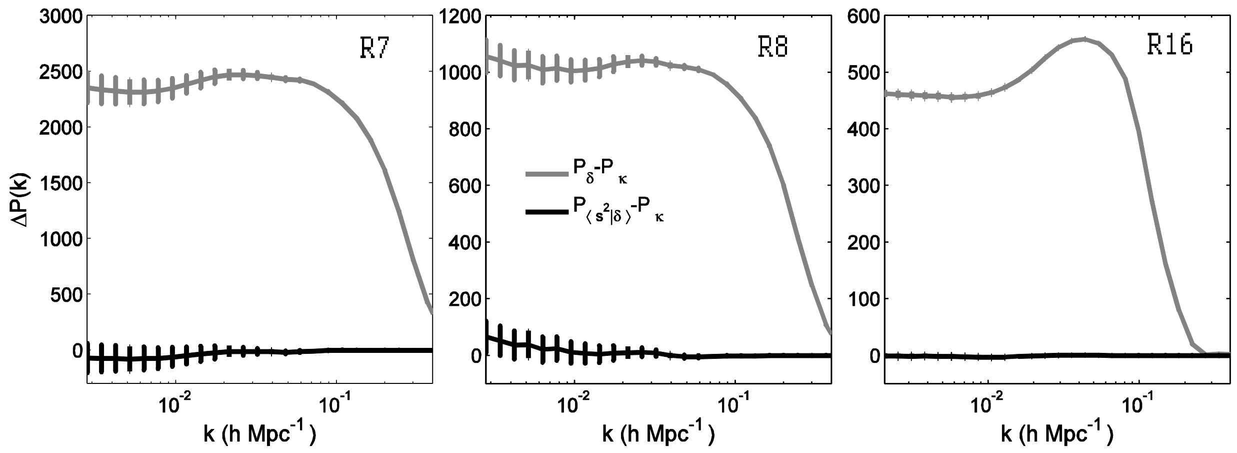
The best unbiased power estimate in the n^{th} column is calculated like so,

$$P_{\langle \boldsymbol{s}^2 | \boldsymbol{\delta} \rangle, n} = \sum_{m=1}^{N} \langle s_m^2 | \boldsymbol{\delta} \rangle B_{mn}.$$

$\langle \boldsymbol{\kappa} \vert \boldsymbol{\delta} \rangle$	clean/shot/kgd_R[r]_sp[02]_[N_{realz}]_fwriteC
$\langle \boldsymbol{s}^2 \vert \boldsymbol{\delta} \rangle$	clean/shot/s2gd_R[r]_sp[02]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\mathbf{Z}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC

B_{mn}	power_spectra/Bmn_R[r]_kappa_l[30]_fwriteR
power/PkBias.m ~ 2000 vectors in (21/12/4) minutes for R7/R8/R16	
$k, \langle P_\delta\rangle, \langle P_\kappa\rangle, \langle P_{\langle\kappa\|\delta\rangle,unbiased}\rangle, \langle P_{\langle\kappa\|\delta\rangle}\rangle$, $\langle P_{\langle s^2\|\delta\rangle}\rangle$, $\langle P_\delta - P_\kappa\rangle$, σ of that average, $\langle P_{\langle s^2\|\delta\rangle} - P_\kappa\rangle$, σ of that average	power_spectra/Pk_kgd_bias_R[r]_l[n_{bins}].txt

The figure below summarizes my findings. In brief, there is a large difference between the power spectrum of the signal and the power spectrum of the data. If the data is replaced with a signal estimate through $\langle \boldsymbol{s}^2|\boldsymbol{\delta}\rangle$ that difference nearly vanishes.



α Scaling

To understand why the signal is being systematically underestimated, we modify $\boldsymbol{\Sigma}_\kappa^{-1} + \alpha\boldsymbol{\Sigma}_\nu^{-1} = \mathbf{W}\boldsymbol{\Lambda}^{(\kappa\nu)}\mathbf{W}^T$ by introducing a scalar factor α.

$\boldsymbol{\Sigma}_\nu^{-1}$	t…/noiseMat_Vec/Sigma_nu_inv_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\boldsymbol{\Sigma}_\kappa^{-1}$	signal/Sigma_kappa_inv_R[r]_[z_{min}]_[z_{max}]_bp62_fwriteC
$\boldsymbol{\delta}$	clean/shot/delta_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\kappa$	clean/shot/delta_kappa_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{\delta}_\eta$	clean/shot/delta_eta_R[r]_[σ_m]_[N_{realz}]_fwriteC
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[30]_fwriteR
clean/WspaceAlpha.m ~ 11hr for diagonalization, 15min for $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$ *(R8)*	
$\mathbf{W}$	g…/clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_alpha[α]_Vectors_fwriteC
$\boldsymbol{\lambda}^{(\kappa\nu)}$	g…/clean/Eigen_W_[σ_m]_R[r]_[z_{min}]_[z_{max}]_bp62_alpha[α]_Values_[.txt,fwriteR]
$\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle$	g…/clean/shot/kgd_R[r]_sp[02]_[1,2]alpha[α]_[N_{realz}]_fwriteC

When reconstructing the signal $\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle$, we have a choice of using $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$ or $\alpha\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$. If the former, use "1" in its filename. If the latter, use "2".

The $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_\kappa$ datasets used in the simulations are identical for all α. The magnitude of noise equals the magnitude of the difference between the data and the signal, $\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2 = 201.2 \pm 0.8$. Prior to cleansing, the magnitude of the signal is $\|\boldsymbol{\delta}_\kappa\|_2 = 321.6 \pm 1.3$. The magnitude of the data is $\|\boldsymbol{\delta}\|_2 = 379.3 \pm 1.3$.

R7

$\boldsymbol{\alpha}$	$\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2$	$\|\boldsymbol{\delta}_\kappa - \langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$ using $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	Drop in noise variance	$\|\boldsymbol{\delta}_\kappa\|_2$	$\|\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$
0.985	201.2 ± 0.8	149.8 ± 0.5	45%	321.6 ± 1.3	288.0 ± 1.3
0.99	201.2 ± 0.8	149.8 ± 0.5	45%	321.6 ± 1.3	287.0 ± 1.3
1.00	201.2 ± 0.8	149.8 ± 0.5	45%	321.6 ± 1.3	284.6 ± 1.3

The $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_\kappa$ datasets used in the simulations are identical for all α. The magnitude of noise equals the magnitude of the difference between the data and the signal, $\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2 = 131.0 \pm 0.6$. Prior to cleansing, the magnitude of the signal is $\|\boldsymbol{\delta}_\kappa\|_2 = 238.7 \pm 1.1$. The magnitude of the data is $\|\boldsymbol{\delta}\|_2 = 272.3 \pm 1.1$.

R8

$\boldsymbol{\alpha}$	$\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2$	$\|\boldsymbol{\delta}_\kappa - \langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$ using $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	Drop in noise variance	$\|\boldsymbol{\delta}_\kappa\|_2$	$\|\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$
0.90	131.0 ± 0.6	104.9 ± 0.4	36%	238.7 ± 1.1	235.5 ± 1.2
0.98	131.0 ± 0.6	102.93 ± 0.4	38%	238.7 ± 1.1	219.2 ± 1.1
0.987	131.0 ± 0.6	102.89 ± 0.4	38%	238.7 ± 1.1	217.8 ± 1.1
0.99	131.0 ± 0.6	102.88 ± 0.4	38%	238.7 ± 1.1	217.3 ± 1.1
1.00	131.0 ± 0.6	102.9 ± 0.4	38%	238.7 ± 1.1	215.4 ± 1.1

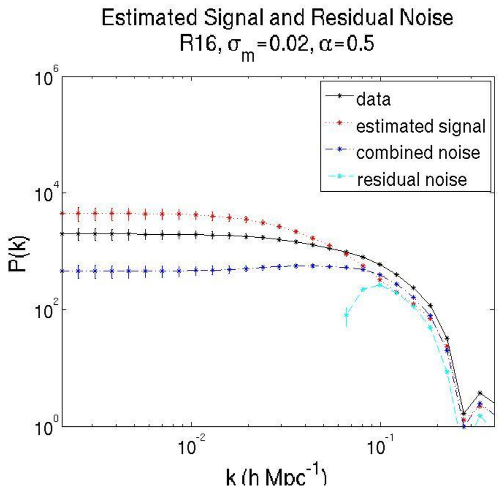
The $\boldsymbol{\delta}$ and $\boldsymbol{\delta}_\kappa$ datasets used in the simulations are identical for all α. The magnitude of noise equals the magnitude of the difference between the data and the signal, $\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2 = 101.5 \pm 0.5$. Prior to cleansing, the magnitude of the signal is $\|\boldsymbol{\delta}_\kappa\|_2 = 77.5 \pm 0.5$. The magnitude of the data is $\|\boldsymbol{\delta}\|_2 = 127.7 \pm 1.1$.

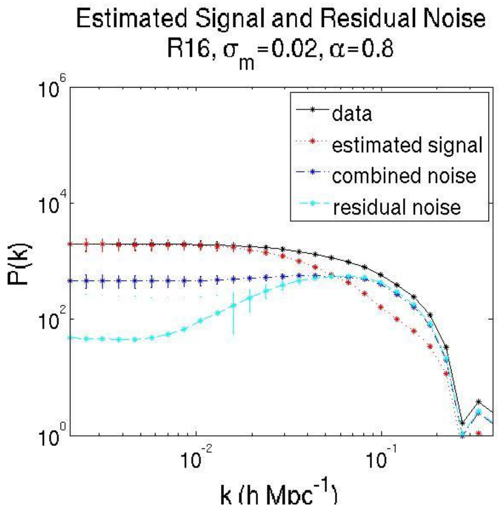
R16

$\boldsymbol{\alpha}$	$\|\boldsymbol{\delta}_\eta + \boldsymbol{\delta}_\zeta\|_2 = \|\boldsymbol{\delta}_\kappa - \boldsymbol{\delta}\|_2$	$\|\boldsymbol{\delta}_\kappa - \langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$ using $\boldsymbol{\Sigma}_\nu^{-1}\mathbf{W}$	Drop in noise variance	$\|\boldsymbol{\delta}_\kappa\|_2$	$\|\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle\|_2$
0.50	101.5 ± 0.5	65.4 ± 0.5	59%	77.5 ± 0.5	104.9 ± 1.0
0.80	101.5 ± 0.5	48.7 ± 0.4	77%	77.5 ± 0.5	73.1 ± 0.6
0.90	101.5 ± 0.5	47.5 ± 0.4	78%	77.5 ± 0.5	66.7 ± 0.6
1.00	101.5 ± 0.5	47.1 ± 0.4	78%	77.5 ± 0.5	61.5 ± 0.5

1.20	101.5 ± 0.5	47.9 ± 0.4	78%	77.5 ± 0.5	53.3 ± 0.4

Keeping the test vectors the same, here's how the residual works out when we scale the noise term when computing $\mathbf{W}$.





The image part with relationship ID rId49 was not found in the file.

The image part with relationship ID rId50 was not found in the file.

For R16, the residual noise appears most unbiased when $\alpha = 0.9$. I repeated the same experiment with R8. The diagonalization of W-space took about half a day, but here's the result. From this it appears that to make this R8 estimator unbiased $0.9 < \alpha < 1$.

The image part with relationship ID rId51 was not found in the file.

Results – Estimated Signal

Each signal estimate $\hat{\boldsymbol{\kappa}}$ will possess its own power spectrum,

$$P_{\kappa,n} = \sum_{m=1}^{N} \hat{s}_m^2 B_{mn}.$$

In this section I will use an ensemble of signal estimates to plot the average power of the recovered signal. By comparing this spectrum to $P_{fid}(k)$ I can quantify the effect of deprojecting the zero-point offsets as a function of k. I can also calculate the covariance matrix of the binned power to see how much estimates in one bin affect those in others.

$\hat{\kappa}_i$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_1e5_fwriteC
$\boldsymbol{\lambda}^{(\kappa)}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$P_{fid}(k)$	base/pkbao_ConvR[r]_fwrite
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_kappa_[n_{bins}]_fwriteR
C_m	power_spectra/Cm_R[r]_kappa_[512]_[1240]_fwriteC
power_spectra/PkOfKappaHat.m ~ < 1 minute	
$k, \langle P(k)\rangle, \sigma_{P(k)}$, $\langle P(k)\rangle - P_{fid}(k)$	power_spectra/PkKappahat_R[r]_[σ_m]_[l,][n_{bins}]_[n_δ]_fwriteR
$\sigma^2(\hat{\boldsymbol{\kappa}})$	power_spectra/SigSqPk_R[r]_[σ_m]_[n_δ]_fwriteR
$\mathrm{Cov}\left(P(k_i), P(k_j)\right)$	power_spectra/CovPkKappahat_R[r]_[σ_m]_[l,][n_{bins}]_[n_δ]_fwriteC

In the table above, the first output files contains 1 rows for every k-bin. If space were less of an issue, I would have written the output like this: $k_n, \langle P(k_n)\rangle, \sigma_{P(k_n)}, \langle P(k_n)\rangle - P_{fid}(k_n)$. The identifier n_δ is a number indicating how many $\hat{\boldsymbol{\kappa}}$ vectors went into the average power at each k value $\langle P(k_n)\rangle$. The standard deviation of the powers that comprise that average are denoted $\sigma_{P(k_n)}$. The final column subtracts the fiducial power spectrum from the recovered signal power spectrum to see how much they differ on various length scales. For consistency I say that in this application,

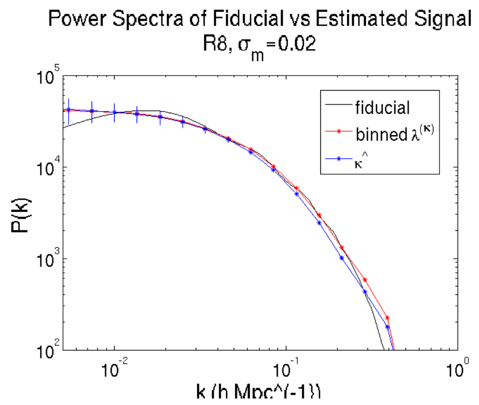
$$P_{fid}(k_n) = \sum_{m=1}^{N} \lambda_m^{(\kappa)} B_{mn}.$$

The second output table contains the integrated power over all length scales. The ith column contains the integral (but really a sum) for the ith solution of $\hat{\boldsymbol{\kappa}}$,

$$\sigma^2(\hat{\boldsymbol{\kappa}}) = \frac{1}{V} \sum_{m=1}^{N} \hat{\kappa}_m^2\, C_m.$$

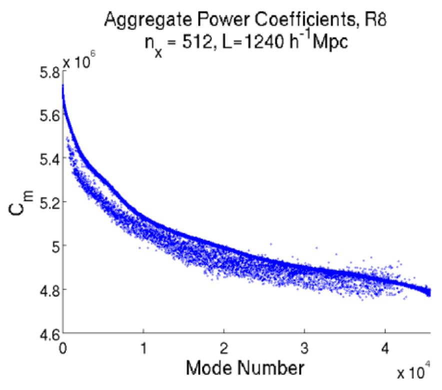
The final output table contains the covariance matrix of the recovered signal power spectrum. Under normal circumstances, there should be no cross-correlations between powers at different length scales. But because of the intersections between the signal and zero-point modes, this relationship breaks down and correlations will be induced.

The code also contains scripts that create the following plots. Let's start with the power spectrum of the estimated signal.



Reconstructing the power spectrum using the B_{mn} coefficients and the $\lambda_m^{(\kappa)}$ variances very closely matches the fiducial power spectrum. This suggests that my process of finding the power of each eigenmode in each Fourier bin is working correctly.

Here's a plot of the $C_m = \sum_{\boldsymbol{k}} |\hat{\boldsymbol{z}}_m(\boldsymbol{k})|^2$ coefficients. While each of the signal eigenmodes has an amplitude of 1, the total Fourier power for each mode is not constant. This shouldn't come as a surprise, however. The higher order modes tend to capture higher frequency behavior. These higher frequencies are the ones most smoothed by the survey window, reducing their amplitudes. As such, the sum of all the Fourier amplitudes should decrease somewhat for the modes with higher frequencies and that's essentially what we observe here.

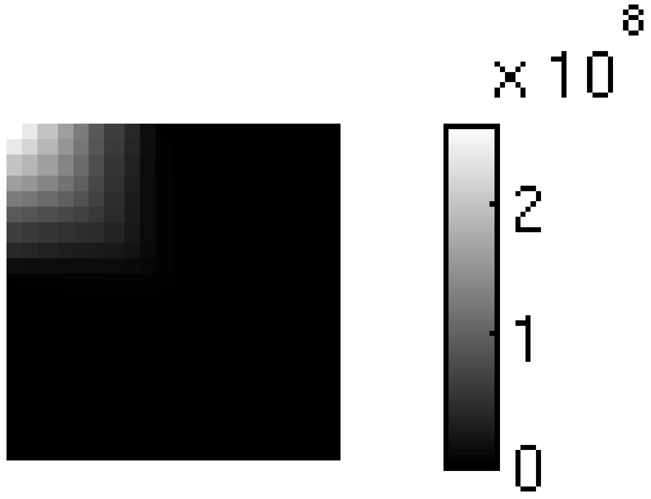


The integrated power shows a similar decrease,

$$\sigma^2\left(\lambda^{(\kappa)}\right) = 9.799$$
$$\sigma^2(\hat{\boldsymbol{\kappa}}) = 8.015 \pm 0.102.$$

The stated error is one standard deviation of an ensemble of 43 $\hat{\boldsymbol{\kappa}}$ solutions.

The covariance matrix of the reconstructed signal powers $\mathrm{Cov}\left(P(k_i), P(k_j)\right)$ is shown below in grayscale,



We can compare the power spectrum of the raw data (i.e. with zero-point and shot noise) with that of the estimated signal. These routines use an adapted form of the old code,

```
power_spectra/PkOfKappaHat2.m ~ < 1 minute
```

My first approach is to find the zero-point noise variance in each dimension of signal-space. An individual noise vector in signal-space equals $\boldsymbol{d}_\eta = \mathbf{M}\boldsymbol{\mathfrak{d}}_\eta$ where $\mathbf{M} = \mathrm{Z}^\mathrm{T}\mathrm{U}$. The variance of the noise along the x^{th} dimension of signal-space should be

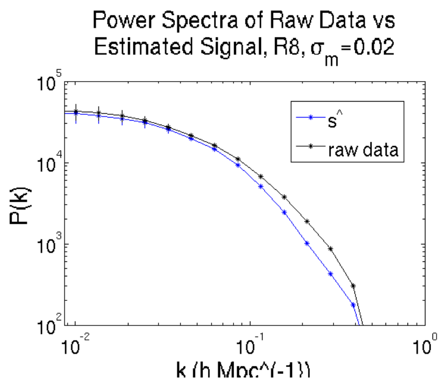
$$\mathrm{Var}\left(d_\eta[x]\right) = \mathrm{Var}\left(\sum_{i=1}^{N} \mathrm{M}_{x,i}\mathfrak{d}_\eta[i]\right) = \sum_{i=1}^{N} \mathrm{M}_{x,i}^2 \,\mathrm{Var}\left(\mathfrak{d}_\eta[i]\right) = \sum_{i=1}^{N} \mathrm{M}_{x,i}^2 \lambda_i^{(\eta)}.$$

By a similar argument, $\boldsymbol{d}_\zeta = \mathrm{Z}^\mathrm{T}\boldsymbol{\delta}_\zeta$. It follows that

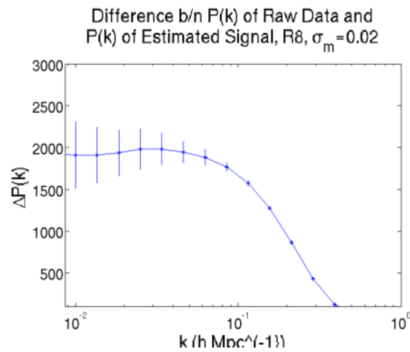
$$\mathrm{Var}(\boldsymbol{d}) = \mathrm{Var}\left(\boldsymbol{d}_\kappa + \boldsymbol{d}_\eta + \boldsymbol{d}_\zeta\right)$$
$$= \mathrm{Var}(\boldsymbol{d}_\kappa) + \mathrm{Var}\left(\boldsymbol{d}_\eta\right) + \mathrm{Var}\left(\boldsymbol{d}_\zeta\right)$$

$$\mathrm{Var}(d[x]) = \lambda_x^{(\kappa)} + \sum_{i=1}^{N} \mathrm{M}_{x,i}^2 \lambda_i^{(\eta)} + \sum_{i=1}^{N} \mathrm{Z}_{i,x}^2 \frac{1}{\langle n_i \rangle}.$$

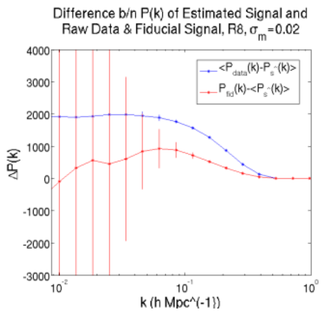
At this point I should be able to plot the spectrum using the $\mathrm{B_{mn}}$ coefficients derived from the signal eigenmodes.



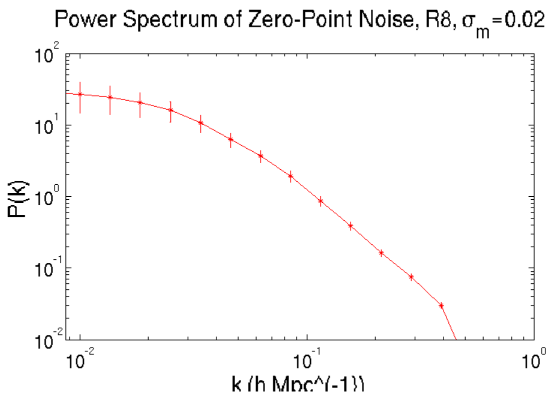
The difference on small scales is likely due to the shot noise.



The following plot contains the same information as above in blue. In red, the n^{th} data point equals $P_{fid}(k_n) - \langle P_{\hat{s}}(k_n) \rangle$. The bump indicates the estimated signal loses power relative to the fiducial at intermediate redshifts, but overall the difference for these 43 vectors is consistant with zero. The spread is gigantic, though, at about 10^4 at the lowest k.



This plot indicates that the signal estimator is largely unbiased since $\langle P_{fid}(k) - P_{\hat{s}}(k)\rangle$ is consistant with zero. There is some departure from zero at intermediate length scales, thought, so there is a possibility that bias exists there. There may be little we can do about this, but it is still improving the signal estimate so our gains outweigh our losses.



The scale of this noise spectrum appears to be off by at least a couple orders of magnitude so I felt it worthwhile to try plotting it again in two different ways to enhance our confidence in it. Rather than do it empirically off a collection of $\boldsymbol{\delta}_\eta$ (actually $\boldsymbol{d}_\eta$), I choose to do it throught a collection of the noise eigenvalues. I must be careful to multiply the noise eigenvalues by σ_m^2.

Z	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwriteC
U	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwrite
$\boldsymbol{\lambda}^{(\eta)}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Values_fwrite
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_kappa_[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_eta_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_eta_[n_{bins}]_fwriteR
power_spectra/NoiseSpectrum.m ~	

My first approach is to find the zero-point noise variance in each dimension of signal-space. An individual noise vector in signal-space equals $\boldsymbol{d}_\eta = \mathbf{M}\boldsymbol{\mathfrak{d}}_\eta$ where $\mathbf{M} = \mathrm{Z}^\mathrm{T}\mathrm{U}$. The variance of the noise along the x^{th} dimension of signal-space should be

$$\mathrm{Var}\big(d_\eta[x]\big) = \mathrm{Var}\left(\sum_{i=1}^{N} \mathrm{M}_{x,i}\mathfrak{d}_\eta[i]\right) = \sum_{i=1}^{N} \mathrm{M}_{x,i}^2 \,\mathrm{Var}\big(\mathfrak{d}_\eta[i]\big) = \sum_{i=1}^{N} \mathrm{M}_{x,i}^2 \lambda_i^{(\eta)}.$$

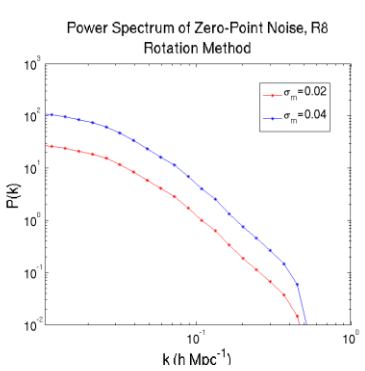
At this point I should be able to plot the spectrum using the B_{mn} coefficients derived from the signal eigenmodes.

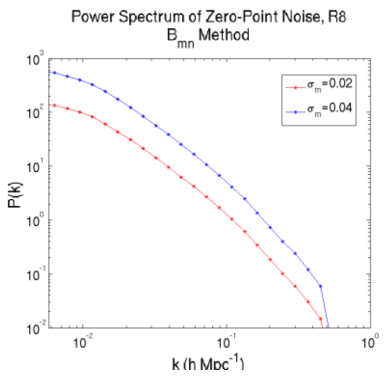
The second method requires that I solve for the B_{mn} coefficients for the noise and plot the spectrum above just as I have for the signal, but swapping out $\boldsymbol{\lambda}^{(\kappa)}$ for $\sigma_m^2\, \boldsymbol{\lambda}_{\sigma_m=1}^{(\eta)}$.

$\mathbf{U}$	noiseMat_Vec/Eigen_DR6_R[r]_[z_{min}]_[z_{max}]_beta_0p62_Vectors_fwrite
$k_x, k_x, k_x, \lvert\boldsymbol{k}\rvert, sd$	cells_grids/kg_[n_x]_[L]_fwriteR
$Cell\ ID, gridbox, fraction$	cells_grids/cgf_R[r]_[z_{min}]_[z_{max}]_[β_{SPEC}]_[512]_[1240]_fwriteR
ind	base/CMATLAB_fftmap_[512]_fwriteC
$Cellcg$	base/Cellcg_R[r]_[512]_[1240]_fwriteR
power_spectra/FFTEigenmodes.m ~ 2.54 min per R8 mode per processor ~ 2 days for R8 on 40 processors ~ 22,200 R8 modes per day on 40 processors	
B_{mn}	power_spectra/coef/Bmn[n]_R[r]_eta_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/coef/Bmn[n]_R[r]_eta_[n_{bins}]_fwriteR
C_m	power_spectra/coef/Cm[n]_R[r]_eta_[512]_[1240]_fwriteC

Finally, I double the magnitude of the zero-points from $\sigma_m = 0.02$ to $\sigma_m = 0.04$ with the expectation that the overall noise power should quadruple.

The result from the rotation method is identical to the one I got empirically. However, the one I get from the B_{mn} coefficients in noise-space give the same shape with a higher magnitude. The two match up fairly well at small scales (where there are plenty of $\boldsymbol{k}$-vectors in each bin) and diverge only around $k \cong 0.05 = 10^{-1.3}$.

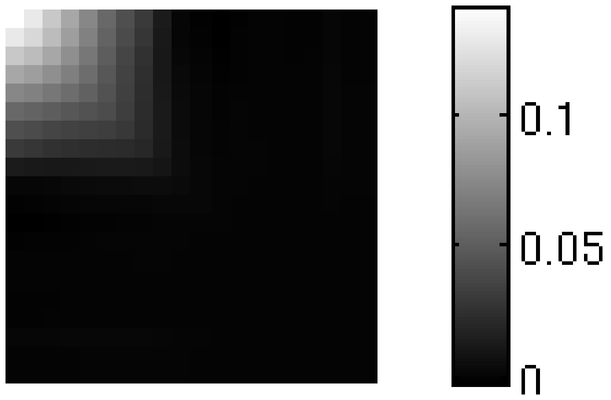




I define the dimensionless variance as

$$\frac{\mathrm{Cov}\left(P(k_i), P\left(k_j\right)\right)}{\langle P(k_i)\rangle\langle P\left(k_j\right)\rangle}.$$

Here's how the plot comes out if I include all of the modes.



The correlations between the large scale k-modes is understandable because of the finite volume size. As we add more modes into the volume they become increasingly dependent. A maximum of about 0.15 is very good because we expect because of the survey window that these large scale modes would be highly correlated.

For a Gaussian power spectrum we would expect this to go to a constant of 2 or 3 along the diagonal after you normalize it. It's unclear why the diagonal doesn't go to a constant.

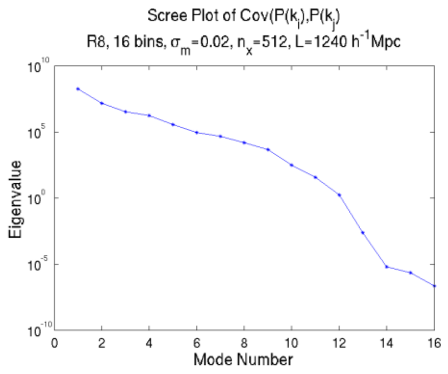
According to Mark, the covariance matrix of the power spectrum goes as something like

$$C_{ij} = \frac{2P_{ij}}{n_{\boldsymbol{k}}}\delta_{ij}.$$

Perhaps the best way to look at the statistic, though, is through something like

$$\mathrm{Corr}_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}}.$$

If I limit myself to only consider the 16 of 200 k-bins that contain more than 200 vectors, then the scree plot of $\mathrm{Cov}\left(P(k_i), P(k_j)\right)$ looks like the figure below. Almost all of the signal is contained in the first mode. Things seem relatively Gaussian.



Results – With Noise Power

In this section I overplot the power spectra of the zero-point noise and raw data to see how they compare to the power spectrum of the signal itself. To do this, I will rotate my test vectors $\boldsymbol{\delta}_\eta$ and $\boldsymbol{\delta}$ into their signal-space coefficients and plot them as I did the estimated signal,

$$\begin{aligned} \boldsymbol{d}_n &= \mathbf{Z}^T \boldsymbol{\delta}_\eta, \\ \boldsymbol{d} &= \mathbf{Z}^T \boldsymbol{\delta}. \end{aligned}$$

The power in the nth bin due to a single vector should then be

$$P_{\eta,n} = \sum_{m=1}^{N} n_m^2 B_{mn}, \qquad P_{\Gamma,n} = \sum_{m=1}^{N} m_m^2 B_{mn},$$

while the integrated power will be

$$\sigma^2(\boldsymbol{d}_n) = \frac{1}{V}\sum_{m=1}^{N} n_m^2\, C_m, \qquad \sigma^2(\boldsymbol{d}) = \frac{1}{V}\sum_{m=1}^{N} m_m^2\, C_m.$$

$\hat{\kappa}_i$	clean/shot/kappahat_given_delta[n]_R[r]_[σ_m]_1e5_fwriteC
$\boldsymbol{\lambda}^{(\kappa)}$	signal/Eigen_kappa_R[r]_[z_{min}]_[z_{max}]_bp62_Values_fwriteR
$P_{fid}(k)$	base/pkbao_ConvR[r]_fwrite
$\boldsymbol{d}$	clean/shot/d_R[r]_[σ_m]_[N_{realz}]_fwriteC
$\boldsymbol{d}_n$	clean/shot/dn_R[r]_[σ_m]_[N_{realz}]_fwriteC
B_{mn}	power_spectra/Bmn_R[r]_kappa_l[n_{bins}]_fwriteR
B_{mn}	power_spectra/Bmn_R[r]_kappa_[n_{bins}]_fwriteR
C_m	power_spectra/Cm_R[r]_kappa_[512]_[1240]_fwriteC
power_spectra/PkOfKappaHat2.m ~ < 1 minute	
$k, \langle P(k) \rangle, \sigma_{P(k)}, \langle P(k) \rangle - P_{fid}(k)$	power_spectra/PkKappahat_R[r]_[σ_m]_[l,][n_{bins}]_[n_δ]_fwriteR
$\sigma^2(\hat{\boldsymbol{\kappa}})$	power_spectra/SigSqPk_R[r]_[σ_m]_[n_δ]_fwriteR
$\mathrm{Cov}\left(P(k_i), P(k_j)\right)$	power_spectra/CovPkKappahat_R[r]_[σ_m]_[l,][n_{bins}]_[n_δ]_fwriteC

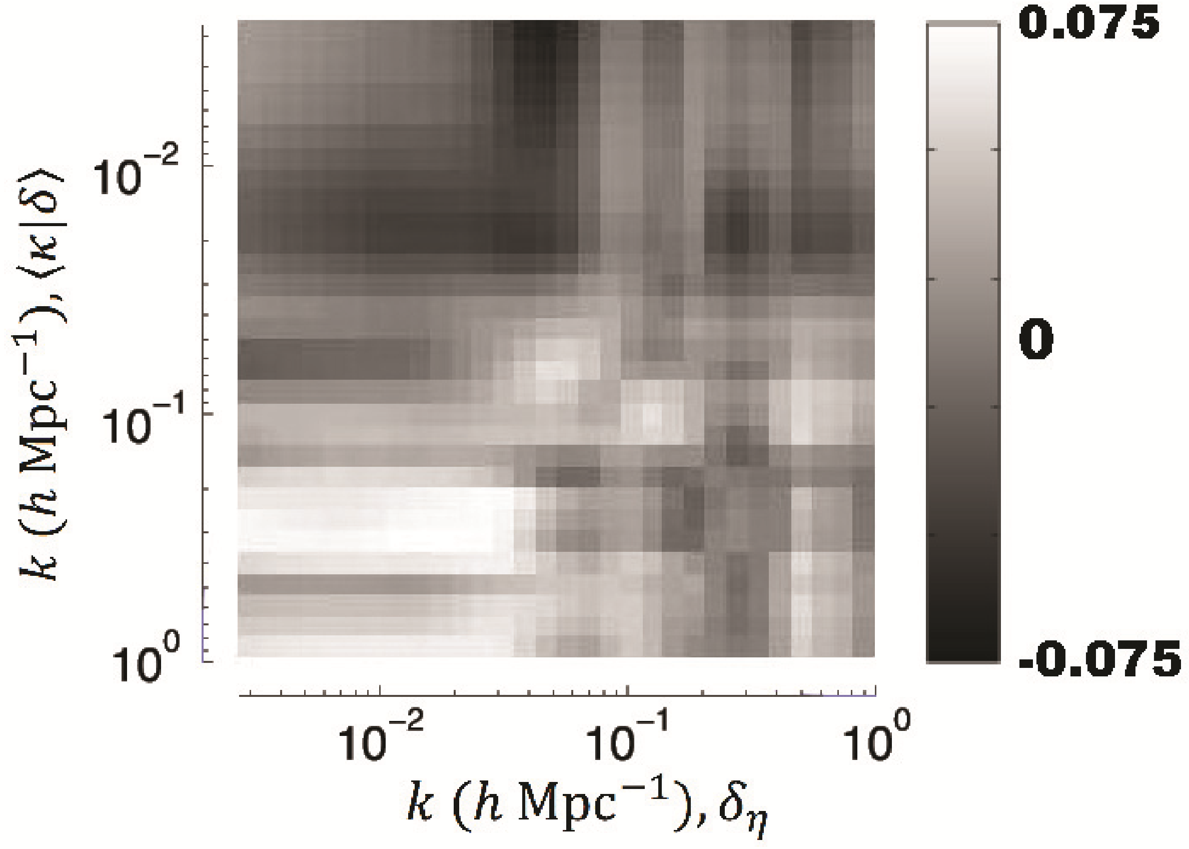
Cross-Correlations b/n Estimated Signal and Zero-Point Noise

The following code investigates the cross-correlations between the power spectra of the zero-point noise and estimated signal. The output is a 30x30 text table with correlation values. I explicitly evaluate the correlations as,

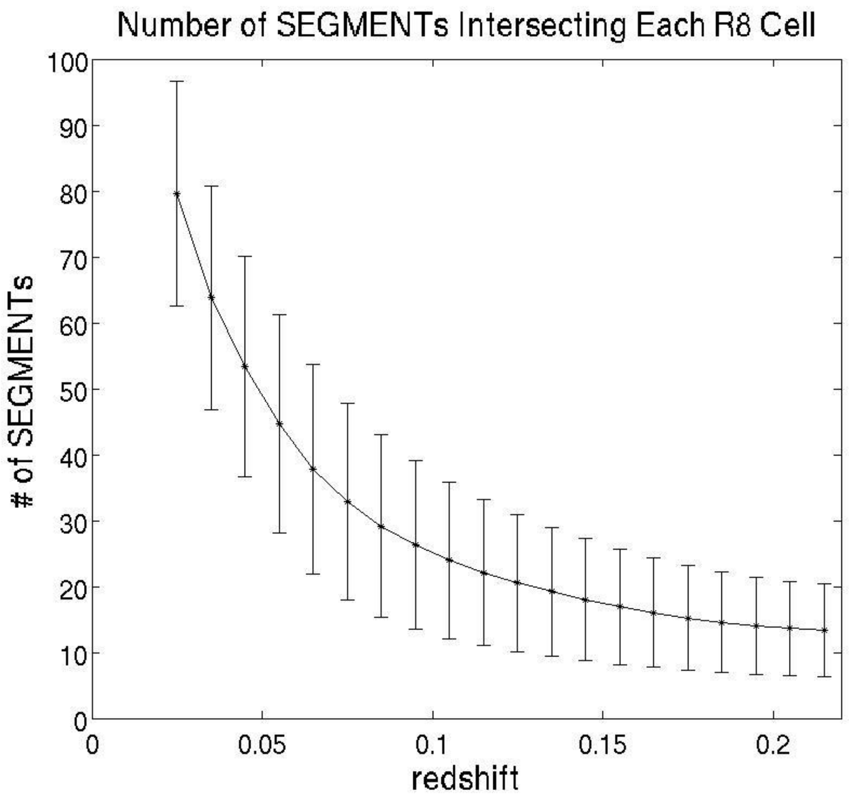
$$\mathrm{Corr}\left(P_{\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle}(k_i), P_{\delta_\eta}(k_j)\right) = \frac{E\left[\left(P_{\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle}(k_i) - E\left[P_{\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle}(k_i)\right]\right)\left(P_{\delta_\eta}(k_j) - E\left[P_{\delta_\eta}(k_j)\right]\right)\right]}{\sigma_{P_{\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle}(k_i)}\sigma_{P_{\delta_\eta}(k_j)}}.$$

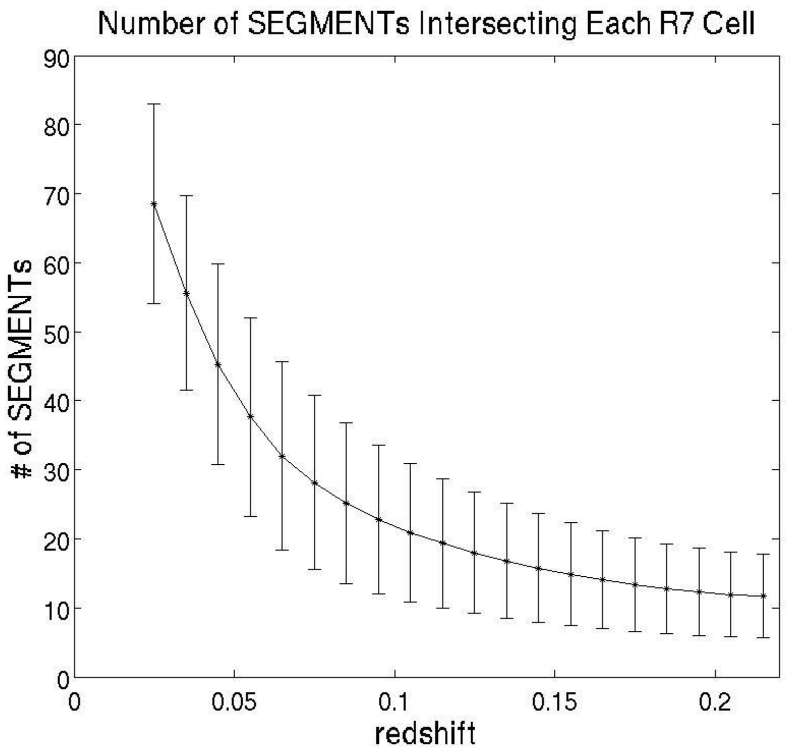
$\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle$	`clean/shot/kgd_R[`r`]_sp[02]_[`N_{realz}`]_fwriteC`
$\boldsymbol{\delta}_\eta$	`clean/shot/delta_eta_R[`r`]_[`σ_m`]_[`N_{realz}`]_fwriteC`
$\mathbf{Z}$	`signal/Eigen_kappa_R[`r`]_[`z_{min}`]_[`z_{max}`]_beta_0p62_Vectors_fwriteC`
B_{mn}	`power_spectra/Bmn_R[`r`]_kappa_l[`n_{bins}`]_fwriteR`
clean/SNcrossCorr.m ~ 20 min or less	
$\mathrm{Corr}(P_{\langle\boldsymbol{\kappa}\|\boldsymbol{\delta}\rangle}, P_{\delta_\eta})$	`clean/Corr_Pkgd_Pdeltaeta_R[`r`]_sp[02]_[`n_{bins}`].txt`

The first 5 rows and columns on the bottom and right relate to the scale size of the spheres. The values of the correlations are represented in gray scale. Each row contains the correlation coefficients between the power in a particular k-band of $P_{\langle\boldsymbol{\kappa}|\boldsymbol{\delta}\rangle}$ and the power in all of the k-bands of P_{δ_η}. The figure below is for R7 when $\sigma_m = 0.02$.



For plotting, there is netplotlib in Python, sm (supermongo) and R.


Number of SEGMENTs Intersecting Each R8 Cell
# of SEGMENTs
redshift
0
10
20
30
40
50
60
70
80
90
100
0
0.05
0.1
0.15
0.2


Number of SEGMENTs Intersecting Each R7 Cell
# of SEGMENTs
redshift
0
10
20
30
40
50
60
70
80
90
0
0.05
0.1
0.15
0.2